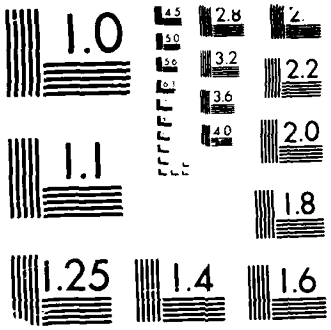
AD-A193 302 AUTOMATIC OMEGA STATION AND LOP (LINE OF POSITION) 1/2
SELECTION(U) DEFENCE RESEARCH ESTABLISHMENT OTTAWA
(ONTARIO) J C MCMILLAN SEP 87 DREO-976

UNCLASSIFIED F/G 17/7.2 NL



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS 1963 A

②

National Defence Défense nationale

DEFENSIO PER SCIENTIAM

AUTOMATIC OMEGA STATION AND LOP SELECTION

by

J.C. McMillan

DEFENCE RESEARCH ESTABLISHMENT OTTAWA
REPORT NO. 976

Canada

September 1987
Ottawa

National Defence Défense nationale

AUTOMATIC OMEGA STATION AND LOP SELECTION

by

J.C. McMillan
Electromagnetics Section
Electronics Division

DEFENCE RESEARCH ESTABLISHMENT OTTAWA
REPORT NO. 976

PCN
041LJ

September 1987
Ottawa

ABSTRACT

Omega is a worldwide navigation system based on 8 VLF transmitting stations dispersed widely around the globe. The phase difference between the signals of any two Omega stations can provide a line of position (LOP) on the earth's surface. Two such LOPs intersect to provide a position fix. Generally several LOPs are selected to improve the accuracy and reliability of a fix, either by using a least squares technique, or as in MINS (Marine Integrated Navigation System) a more sophisticated Kalman filter technique. The resulting accuracy and reliability depends on many factors such as geometry and signal strength. This report describes these factors and how they can be evaluated. It also describes a set of algorithms used by MINS to automatically select a best choice of 5 Omega stations and 4 LOPs. This includes a modal interference predictor and multi-LOP position accuracy calculations for both least squares and Kalman filter solutions.

RÉSUMÉ

OMEGA est un système mondial de navigation, basé sur huit postes de transmission répartis autour du globe. La différence de phase entre les signaux de deux postes donne un ligne de position (LDP) sur la surface de la terre. L'intersection de deux LDPs donne une position. Généralement plusieurs LDPs sont choisies pour améliorer l'exactitude et la fiabilité d'un relèvement, soit par l'utilisation d'une technique des moindres carrés, ou comme pour MINS (Système Intégré de Navigation Maritime) par la technique plus sophistiquée de filtrage de Kalman. L'exactitude et fiabilité qui en résulte dépend de plusieurs facteurs, tels que la géométrie et la puissance du signal. Ce rapport décrit ces facteurs et leur évaluation. Il décrit également l'ensemble des algorithmes employé par MINS pour sélectionner automatiquement le meilleur ensemble de cinq postes OMEGA et quatre LDPs. Ceci inclus une prédiction d'interférence modale et un calcul multi-LDP de la dilution de la précision géométrique.

TABLE of CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.0 BACKGROUND

Although the Omega automation method described in this paper is applicable to any integrated navigation system that uses Omega, or to any automatic Omega receiver, it was developed at DREO for use in MINS (Marine Integrated Navigation System).

The MINS is a Kalman filter based optimally integrated system designed to improve the navigation accuracy, efficiency and reliability within the Canadian Navy, by combining the output from all navigation sensors onboard each vessel to automatically produce one best estimate of position velocity and other parameters of interest.

The MINS concept and system design were developed at DREO over the period 1980-1987. The initial simulation study is summarised in reference [1], at which time only the gyrocompass, speedlog and Omega were considered for integration. In 1981 Loran-C was added, still at the simulation level. This design, along with some of the analysis that was used for its development, is described in reference [2].

A laboratory development model was then assembled, using an LSI 11/23 general purpose computer with an RSX operating system and FORTRAN as the language. The structure of this DREO developed software package is briefly described in reference [3]. The interfaces were supplied by JMR Instruments Canada Ltd. The sensors were:

1-Sperry Mk 23 Mod C-3 gyrocompass
2-Sperry SRD-301 Doppler speedlog
3-AN/SRN 12 Omega receiver
4-Internav LC204 Loran-C receiver

This lab model was tested on Canadian Forces research vessel CFAV Endeavour, from April to November 1982 and May to August 1983. The 1982 trial supplied data to debug and refine the interfaces, the navigation software and most importantly, the optimal integration software. This also led to the development of error detection methods for handling spurious speedlog data, Omega lane slips, Loran cycle selection errors etc.. Reference [4] describes some of this work.

The Canadian Navy acquired MX1105 Omega-Transit receivers to replace the archaic SRN-12s and add Transit capability. In 1983 Transit capability was added to create MINS-B. This improved system was tested from May to August 1983. The final 1983 trial, using a Maxiran shore based high frequency reference system, proved the high accuracy potential of the MINS-B algorithms. The addition of Transit and some 1983 sea trial results are described in reference [5]. The sensors for this and subsequent trials were:

1-Sperry Mk 23 Mod C-3 gyrocompass
2-Sperry SRD-301 doppler speedlog

3-Magnavox MX1105 Transit-Omega receiver
4-Internav LC204 Loran-C receiver

In 1983 JMR (which became EDO Canada later that year) was contracted to build an ADM, implementing the DREO MINS-B software on a Motorola 68000 based microcomputer. This required the translation of the software to the "C" language (because of compiler availability at that time). This MINS-B ADM, along with the DREO laboratory MINS, were tested together on both the CFAV Endeavour on the west coast and the HMCS Cormorant on the east, in 1984. The major verification sea trial was in October 1984 on the Endeavour, with a Syledis reference system. The results of this trial proved the accuracy performance of the MINS-B ADM and are reported in detail in reference [6].

In 1985 the U.S. Navy acquired an upgraded MINS-B ADM for evaluation. NADC conducted sea trials (for NAVSEA) with this unit in 1985, 86 and 87 on the USS Reasoner and on the Vanguard. In 1986 DREO incorporated GPS into the MINS filter. In 1986 and 1987 DREO added many software enhancements to MINS to improve the operability, the accuracy and the failure modes. Reference [7} describes some of the sensor error and failure handling techniques used in MINS. At the same time EDO improved the hardware significantly, going to a 68020 processor with a math coprocessor, reducing the size of the interface hardware and separating the control/display unit from the electronics unit.

In 1987 EDO was contracted to build an EDM, which was to later be modified to become the first preproduction unit.

-Formal sea trials by DREO:
- lab model 1982 & 1983
- ADM 1984
- EDM 1986

2.0 PROBLEM STATEMENT

This report deals with the problem of automating the selection of Omega stations and LOPs for MINS, by maximising the expected accuracy and reliability. There are a total of 8 Omega stations to choose from, but the receiver can only lock onto 7 at a time, and in practice it is very seldom that more than 5 signals are actually received. In MINS we therefore model only 5 Omega phase errors at a time, which means that MINS can process up to 4 LOPs at a time (formed from linearly independent pairs of these phases). The problem is therefore to select the best choice of 4 LOPs using 5 stations. This must be done at system initialisation, and from time to time thereafter (once an hour should be sufficient). The relevant information to make this selection that is available or that can be computed is:

1/ geometric dilution of precision
2/ signal to noise ratio
3/ modal interference
4/ operator deselection
5/ stations tracked by receiver

At initialisation the very best set found should be used, but subsequently it is not worth changing station selection for a marginal improvement in expected accuracy. The practical consideration is that each station used must have its PPC (Phase Propagation Correction) calculated before it can be used, and the PPC task is computationally very burdensome. Rearranging the same stations into new LOPs, however, is very easy and can be done without hesitation. Therefore when "reselecting" every hour, if the previously selected stations are all still available, and the predicted improvement in position accuracy from a new selection (compared to the best set of LOPs that can be formed by using the same stations) is less than say 15%, then the previous station selection should be maintained.

Another important consideration for the Kalman filter is that when a change of station and/or LOPs is made, to avoid loss of information it is important to save the values of the filter states and covariances for any stations that are kept from the previous selection. This will simply involve rearranging the rows and columns of the covariance matrix and elements of the state vector, with reinitialization only for nsw stations.

2.1 SOLUTION APPROACH

The selection process can be formalised by describing it in six stages as follows:

1/ determine which of the eight stations should be considered acceptable for forming LOPs, given:

a- the signals locked onto by the receiver
b- operator deselected stations
c- signals with very low signal to noise ratio
d- signals with high risk of modal interference

2/ estimate how accurate the phase information is likely to be from each acceptable signal by:

a- expressing the expected signal phase errors (in metres) for each signal as a function of the signal-to-noise ratio.
b- determining the expected or measured S/N ratio.

3/ enumerate the possible choices of LOP sets (of at most 4 linearly independent LOPs formed from at most 5 stations) given the stations available from 1 above.

4/ express the expected position accuracy resulting from each choice, by expressing the expected radial position error as a function of the phase errors estimated in 2 above, assuming a least squares solution, using the calculated geometric dilution of precision (GDOP).

5/ evaluate the accuracy for all sets enumerated in 3/ above, using the results of 4/ as the criteria to make a preliminary selection of the 6 or 7 most accurate sets.

6/ apply a reliability criteria to the most accurate sets chosen in 5/ above to ensure that the same station is not used in 3 or more LOPs, (to minimise the effect of temporary station loss) and also, if it is not the initial selection, to apply an efficiency criteria to avoid changing stations for marginal improvement.

These six steps are described in chapters 3 to 8 below, and an example is given in chapter 9.

3.0 STATION ELIMINATION

The first step in selecting Omega LOPs is to determine which stations are, or are likely to be, available. The receiver will indicate which stations are locked onto and what the signal-to-noise (S/N) ratios are (at least the MX1105 used by the Canadian Navy does), and MINS will already have operator deselection information. Thus items 1a and 1b of section 2.1 above are quite trivial. For item 1c it is only necessary to decide upon the minimum acceptable S/N ratio.

3.1 SIGNAL TO NOISE RATIO

As described in reference [8] the MX1105 provides linear receiver S/N ratio in a 100-Hz bandwidth, with a range of 0.00 to 0.99, with 0.00 indicating essentially no signal and .99 indicating a very strong signal. The receiver itself disables the use of any station with a S/N of less than 0.1. For MINS we set the threshold at .20, unless this results in fewer than 5 stations being acceptable, in which case we loosen the threshold to .10 to pick up any marginal stations.

The equivalent dB values for the MX1105 S/N ratio output are listed below (from private communications with Magnavox):

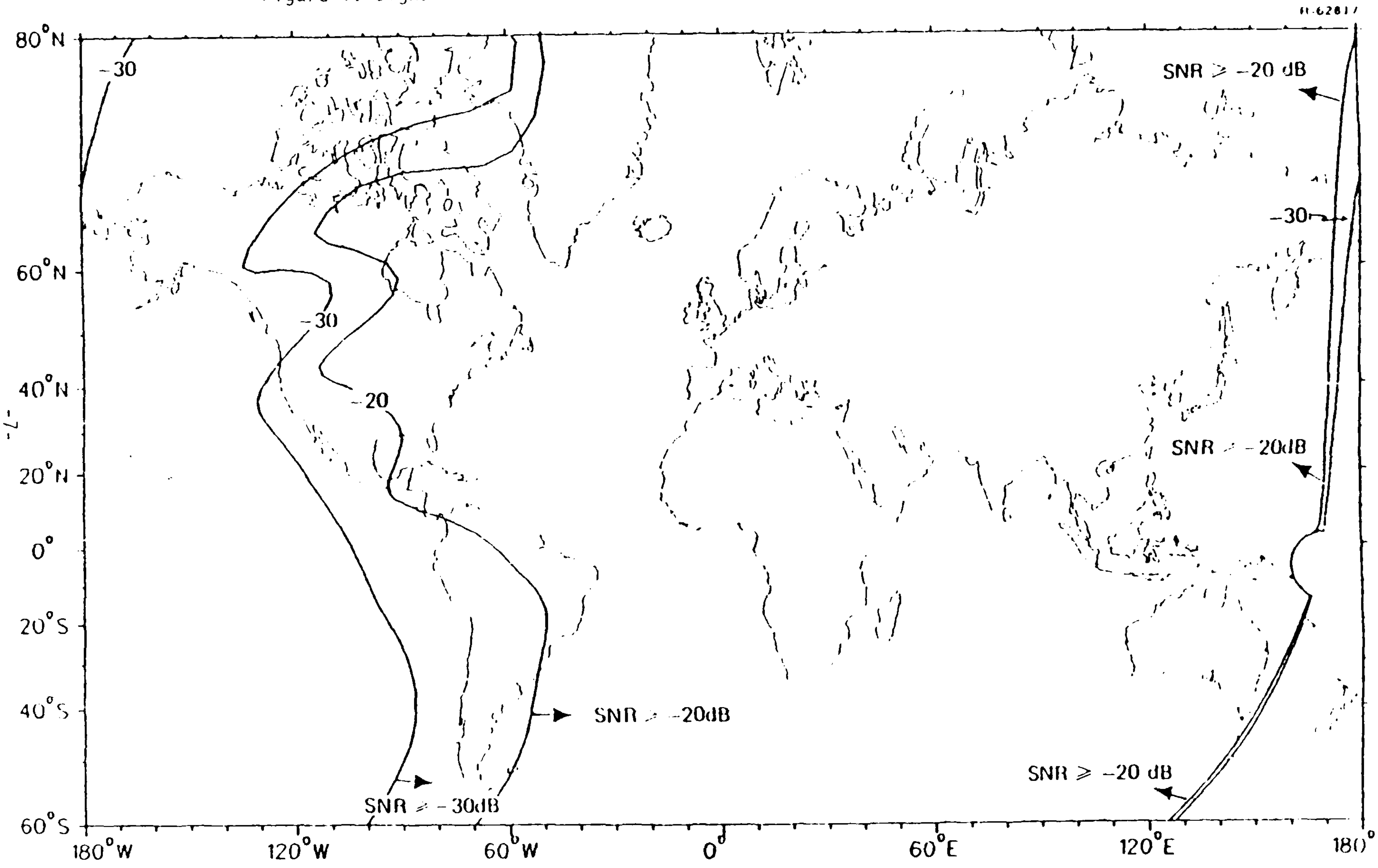
MX1105 Value	dB
0.00	-40.
0.03	-30.
0.10	-19.
0.20	-13.
0.30	- 8.
0.40	- 6.
0.50	- 4.
0.60	- 2.
0.70	- 1.
0.80	3.
0.90	10.
0.95	16.
0.97	20.
0.98	30.
0.99	40.

We also use S/N ratio to ensure that less than 7 stations are used for the enumeration and GDOP calculation, since for 7 stations this is computationally very burdensome. This is done simply by rejecting the station with the lowest S/N ratio if seven stations are still considered acceptable at the end of the station elimination process, which is quite unlikely and in any event still leaves six stations to choose from.

Reference [10] provides maps for each Omega station, illustrating with contours the predicted regions on the earth where the expected S/N ratio will be > -20 dB and where it will be > -30 dB, for 4 different seasons of the year, and 2 different times of day. These are theoretical results, assuming 10 kW transmitter radiated power for each station, and a receiver noise bandwidth of 100 Hz. Figure 1, taken from reference [10], provides an example of these S/N ratio regions. It is for the Liberia Omega station, and predicts the acceptable S/N regions for 18:00 GMT on February (to be used over the January-March period from 12:00 GMT to 24:00 GMT).

This S/N ratio is also a factor in the expected Omega phase error, as described in chapter 4.

Figure 1. Signal to Noise Ratio for Liberia, 1800 GMT, February.



3.2 MODAL INTERFERENCE

The other major consideration when choosing Omega stations is the problem of modal interference, item 1d. This phenomenon is described in references [9] and [10], which also provide an excellent tutorial on the Omega signal propagation characteristics. To understand modal interference, it is necessary to know that the Omega signal is effectively travelling in a waveguide formed by the earth's surface and the D-region of the ionosphere (a steep conductivity gradient between 70 and 100 km.). The ionosphere is significantly affected by the earth's magnetic field (making it anisotropic) and by solar irradiation (which lowers the effective height of the D-region). Although it is intended that the Omega transmitters produce a single mode, it is inevitable that secondary modes are produced. Normally the secondary modes dissipate much more quickly than the primary mode, so that they don't affect the receivers, but this is not always the case.

The magnetic field induced anisotropy causes east-to-west propagating signals to attenuate much more rapidly than west-to-east signals, especially at low geomagnetic latitudes where the horizontal component of the geomagnetic field is strongest. This dissipation also effects the primary mode more than the secondary modes. The result is that in regions west of the transmitter (especially at low latitudes) the primary mode can be dominated by the secondary modes, a situation known as modal interference which prevents receivers from properly locking onto the primary signal.

The effect of solar illumination on the ionosphere is to lower it, thereby decreasing the width of the waveguide. The Omega frequency was chosen to optimise propagation through this daytime waveguide, so it is at night when the ionosphere rises that problems occur. The secondary modes dissipate much more slowly under nighttime conditions, again causing modal interference.

In general there are primarily two contiguous regions of the earth's surface, for each Omega station, where the signal from that station can be expected to suffer from modal interference, which can prevent the receiver from locking onto the desired phase. One region is near the transmitter, where the higher order modes have not had a chance to dissipate yet. The other region is near (and especially to the east of) the antipodal point, where the wrong-way path signal interferes. This is because, as mentioned above, the east-to-west propagation at night of the higher order modes (especially at low geomagnetic latitudes) is less dissipated than the primary mode propagating eastward in daylight (see reference [9] for a very brief physical explanation). For some stations the near field region stretches westward to join with the far field region (Argentina, Liberia, Hawaii and Japan) to make one large region.

The shape of these regions is influenced primarily by the earth's magnetic field and the illumination of the ionosphere by the sun. The solar effect therefore causes some change in shape of the regions as a function of time of day and year. The dominant effect however is geomagnetic, so that it is possible to first predict the constant regions assuming no solar illumination:

the "nighttime model", and then account for the solar effect by calculating the proportion of the path from receiver to station that is in fact illuminated. If a large part of this path (more than 1/3 say) is in darkness and the nighttime model predicts modal interference, then one can assume modal interference, otherwise one can assume no modal interference, unless the range to the station is less than 500. km. where the near field effect is always present, or more than 19,000. km. where the wrong-way path signal interfers.

Reference [9] illustrates these regions on Mercator projection maps of the earth's surface (excluding polar regions). These plots have been reproduced here as figures 2 to 9. The Omega Navigation System center (U.S. Coast Guard) (formerly Omega Navigation System Operations Detail (ONSOD)) has an algorithm for producing these region maps, but unfortunately the computational burden is far in excess of what can be implemented in real time on a microprocessor at this time. Some automatic navigation systems have therefore represented these regions as bit maps, which requires 8 two dimensional arrays, each covering the full range of 180 degrees of latitude and 360 degrees of longitude. For a resolution of only .5 degrees this would require at least 2,000,000 bits, or .25 Mbyte. We have found that this large data base is not necessary, as explained below.

3.2.1 NIGHTTIME MODEL (MAGNETIC EFFECT)

Although these modal interference regions have a fairly complex shape in latitude/longitude coordinates, a little insight reveals that the regions can be thought of as a minimum and maximum range from the station, for any given bearing. In other words a geodesic from the transmitting station to any point on the earth's surface will cross the region's boundary at most twice, first leaving the modal interference region near the transmitter, and then reentering the modal interference region near the antipodal point. (In the four cases where these two regions are joined, the simple and obvious solution is to split the regions into two at their "choke point", as will be demonstrated below.) This concept allows the two dimensional regions to be expressed as reasonably simple one dimensional functions, namely the minimum range and the maximum range from the transmitting station, as functions of bearing from the transmitting station.

This concept was verified by digitising the region boundaries as latitude/ longitude pairs, using the ONSOD maps (figures 2 to 9) on a digitising table. These x-y points had to be converted from Mercator to Cartesian coordinates to obtain the correct latitude and longitude. Then the range and bearing from the transmitting station to each of these positions was calculated. When the resulting ranges were plotted as a function of the bearing, as shown in figures 10 to 17, it was clear that the result was two single valued functions: the minimum and maximum range functions. In several cases, such as the Japan station, the close and far field regions joined. In these cases there was a set of bearings for which all ranges were in the modal interference zone, in which case the range functions had to be completed so that the minimum range exceeded the maximum range. This was easily done without disturbing the continuity of the functions.

These range functions can be approximated by their truncated Fourier series. We found that very good approximation could be achieved by using 6th order Fourier series (for some of the functions a lower order would have been quite adequate, but for uniformity we elected to use 6th order for all functions). To obtain an adequate fit, however, we separated the common minimum range of 500 km. and the maximum range of 19,000 km., to be applied first: If the range is not between these limits then there is no need to evaluate the more complicated range functions. By doing this we can allow our Fourier approximated functions to go smoothly through these hard limits, thereby making a better match between these limits, where it counts. (Fourier approximations have difficulty matching sharp edges.) We did this by "smoothing out" the range functions (figures 10 to 17) where they hit these hard limits before finding their Fourier series. The resulting approximating functions are illustrated in figures 18 to 25, where the 500. km. and 19.000. km. limits have been applied to the Fourier approximations.

Each range function is therefore of the form:

$$\begin{aligned} R = \; & a_0 + a_1\sin\theta + a_2\sin2\theta + a_3\sin3\theta + a_4\sin4\theta + a_5\sin5\theta + a_6\sin6\theta \\ & + b_1\cos\theta + b_2\cos2\theta + b_3\cos3\theta + b_4\cos4\theta + b_5\cos5\theta + b_6\cos6\theta \end{aligned} \tag{1}$$

The values of the coefficients a_i and b_i, in kilometres, are shown in Table I below. The first 8 rows are for the minimum ranges to the 8 stations, in their usual order (Norway, Liberia, Hawaii, North Dakota, La Reunion, Argentina, Australia and Japan). The second 8 rows are coefficients for the maximum range functions.

For computational purposes equation (1) can be significantly improved, to minimise the number of trigonometric function calls. By using multiple angle formulae, it can be shown that the above expression is equivalent to

$$\begin{aligned} R = \; & (a_0 - b_2 + b_4 - b_6) + x(a_1 + 3a_3 + 5a_5) + y(b_1 - 3b_3 + 5b_5) \\ & + xy(2a_2 + 4a_4 + 6a_6) + y^2(2b_2 - 8b_4 + 18b_6) \\ & + x^3(-4a_3 - 20a_5) + y^3(4b_3 - 20b_5) \\ & + x^3y(-8a_4) + y^3x(-32a_6) + y^4(8b_4 - 48b_6) \\ & + x^5(16a_5) + y^5(16b_5) + y^5x(32a_6) + y^6(32b_6) \end{aligned} \tag{2}$$

where

$x = \sin\theta$
$y = \cos\theta$

and where all terms in brackets are constants that can be precalculated. The powers of x and y can be built up with 12 multiplications, and so the evaluation of these range limits requires essentially only 2 trigonometric function and 25 multiplications.

These range functions are then used in the automatic station selection software. To illustrate how well they approximate the original regions illustrated in figures 2 to 9, both the original (Mercator) regions and the DREO approximations (range functions) are expressed in Cartesian latitude/ longitude plots, shown in figures 26 to 41.

It must be kept in mind that the minimum range of 500. km. and the maximum range of 19,000 km. must be applied to these Fourier series to obtain the range functions shown in figures 18 to 25.

TABLE I

FOURIER COEFFICIENTS FOR MODAL INTERFERENCE FUNCTIONS

	a0	a1	b1	a2	b2	a3	b3	a4	b4	a5	b5	a6	b6
A	551	-33	61	-2	2	-18	7	-1	-5	-6	-5	4	-2
B	1372	170	583	-1380	493	-442	-418	319	-378	255	196	-122	96
C	1217	-237	703	-1091	-321	435	-214	421	292	-291	-36	-85	-118
D	563	2	26	-42	30	15	14	-24	-6	4	-0.3	-2	-1
E	773	240	376	-154	200	-108	-43	6	-35	-5	10	-17	-12
F	911	100	182	-344	-399	-217	-141	-146	100	53	183	124	120
G	646	-2	228	-178	18	-27	-154	123	-11	6	95	-75	12
H	1013	-59	629	-418	401	-225	-523	140	5	154	4	-167	63

	a0	a1	b1	a2	b2	a3	b3	a4	b4	a5	b5	a6	b6
A	16642	1392	3764	1619	-1548	-595	-134	312	-325	-430	6	29	73
B	11780	2839	9938	2490	-870	1282	246	602	-529	256	-518	-60	-55
C	12069	2033	9210	764	-3474	-921	-1697	-494	-2038	-349	-750	590	-890
D	18333	2482	5680	1753	-2037	-788	5	143	-282	-524	200	198	192
E	14681	-105	6632	-525	2925	1859	1069	179	582	755	263	328	246
F	14134	-1343	6111	661	3383	2832	1051	811	-387	391	-194	228	78
G	14087	339	4416	-2711	2971	3016	941	-422	218	597	63	304	162
H	17696	4081	11965	722	-2791	298	2940	934	-1344	228	1269	864	-1325

Figure 2. Lat-Long ONSOD Nighttime Modal Interference Model, Norway

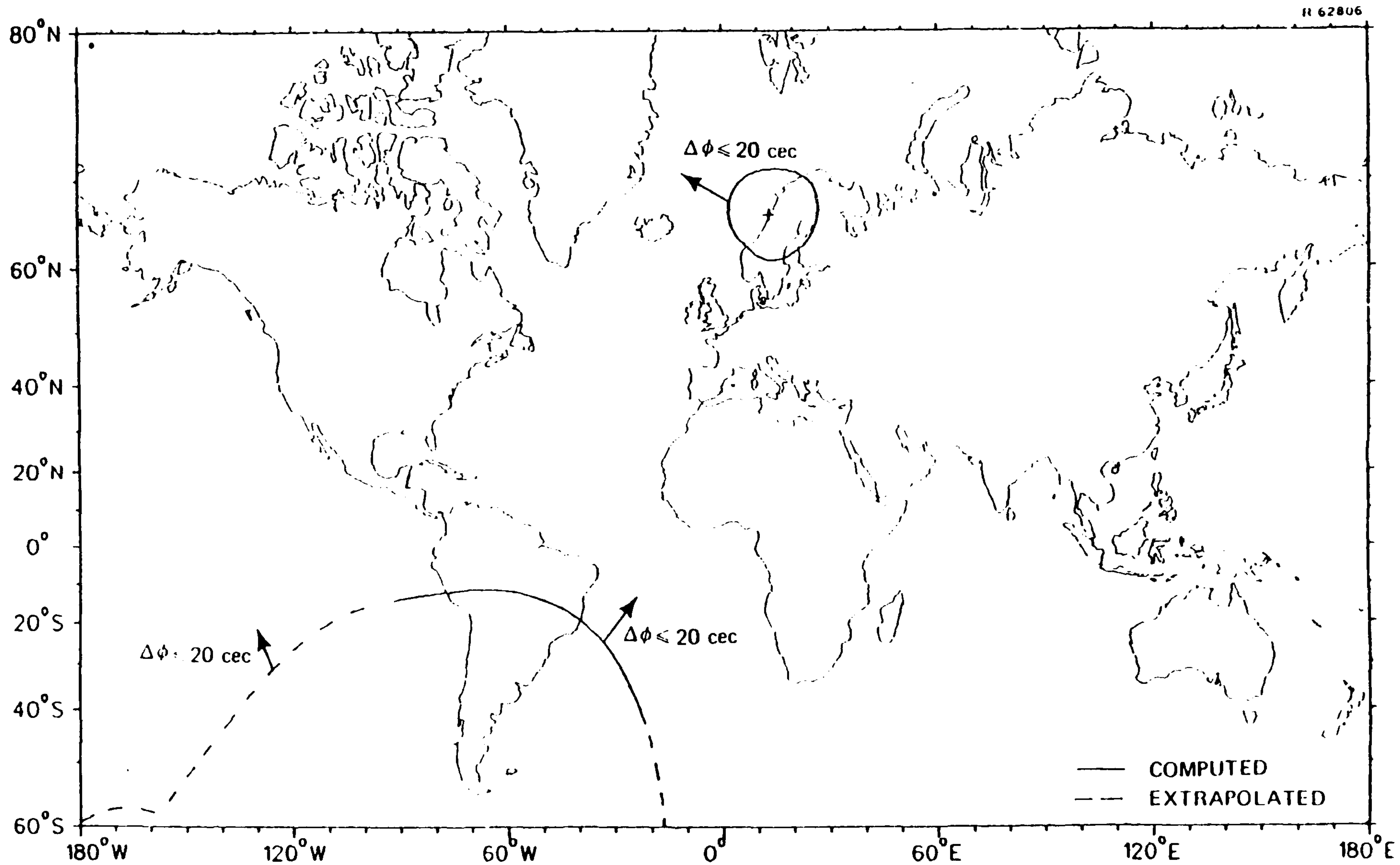


Figure 3. Lat-Long ONSOD Nighttime Modal Interference Model, Liberia

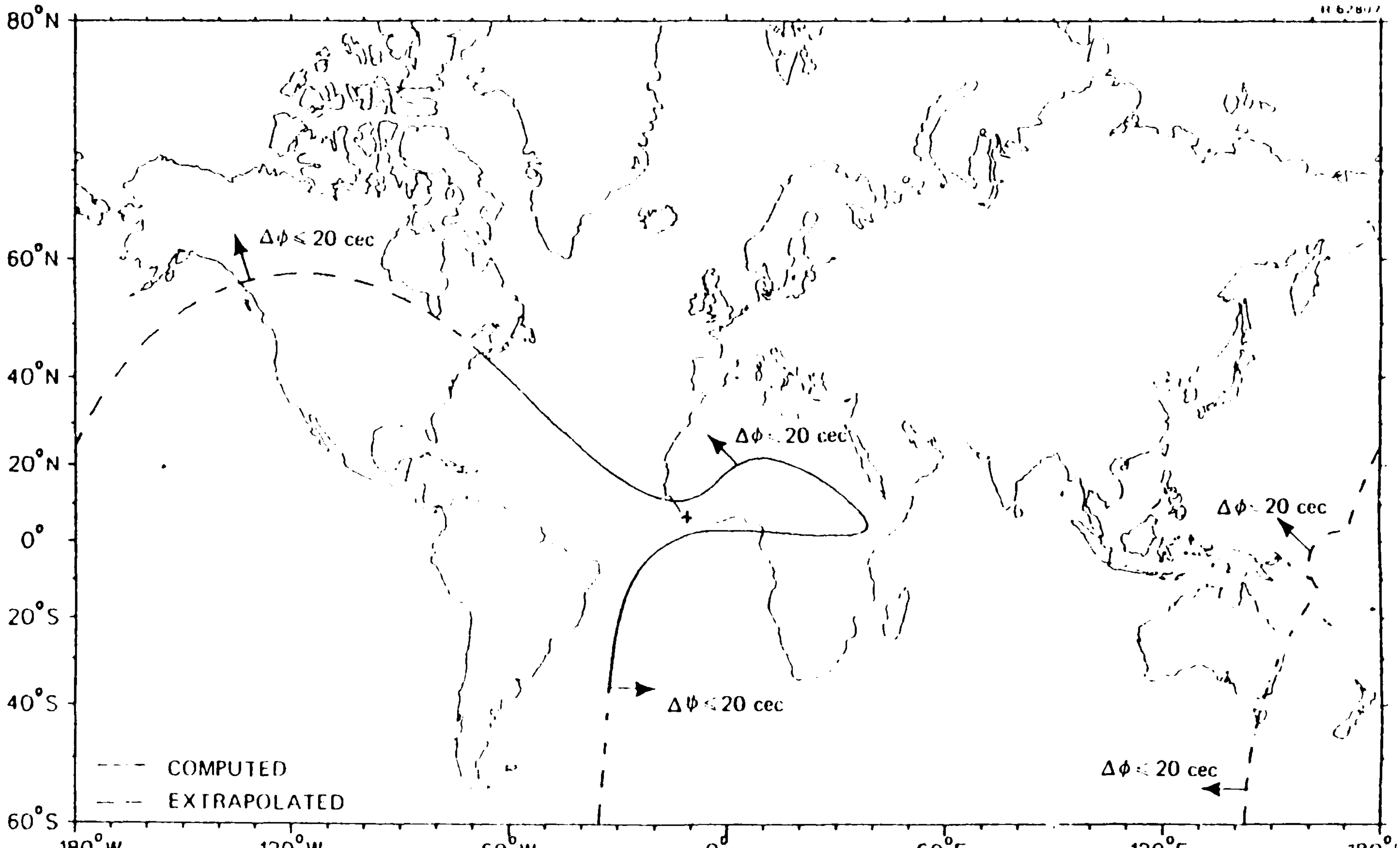


Figure 4. Lat-Long ONSOD Nighttime Modal Interference Model, Hawaii

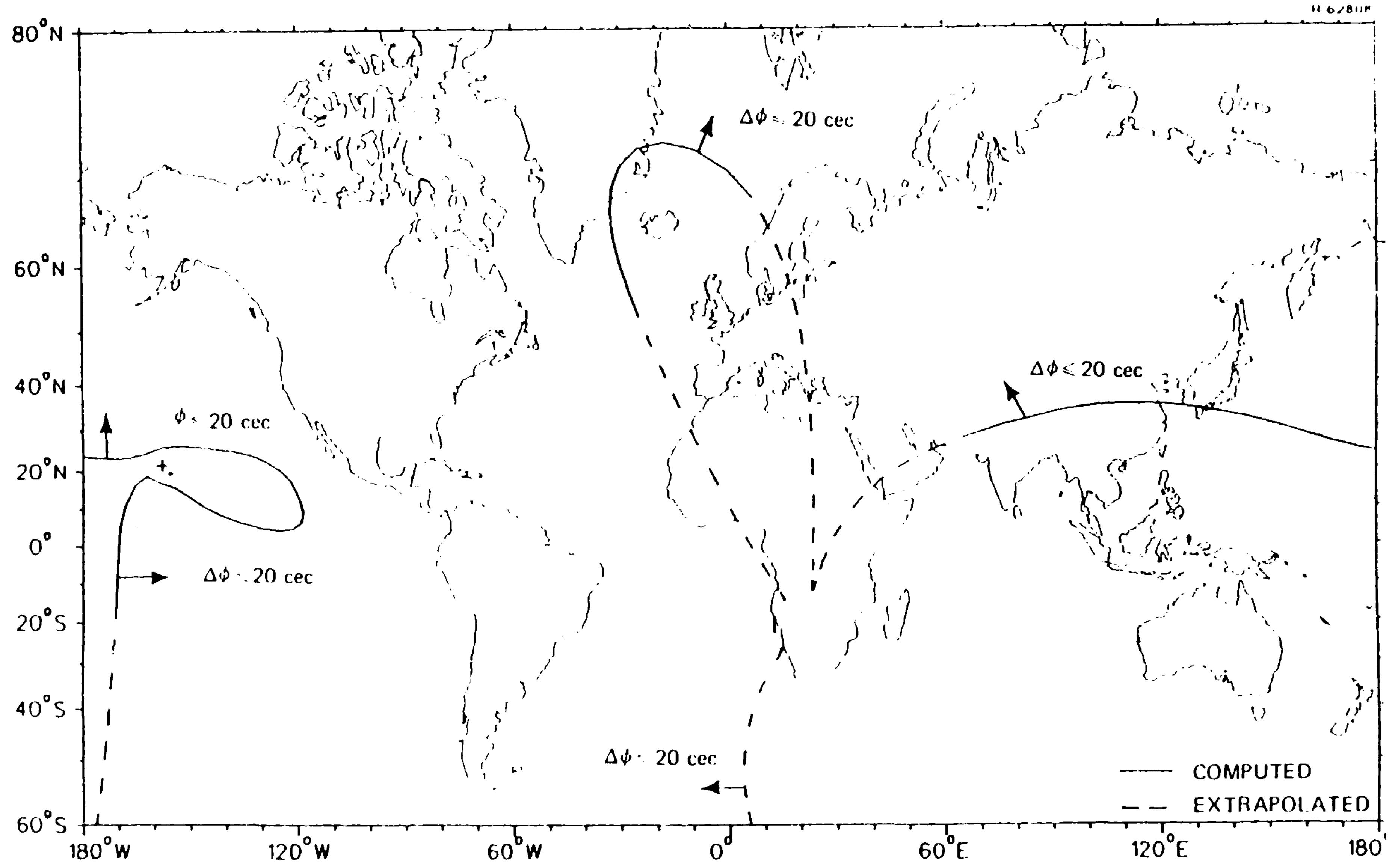


Figure 5. Lat-Long ONSOD Nighttime Modal Interference Model, North Dakota

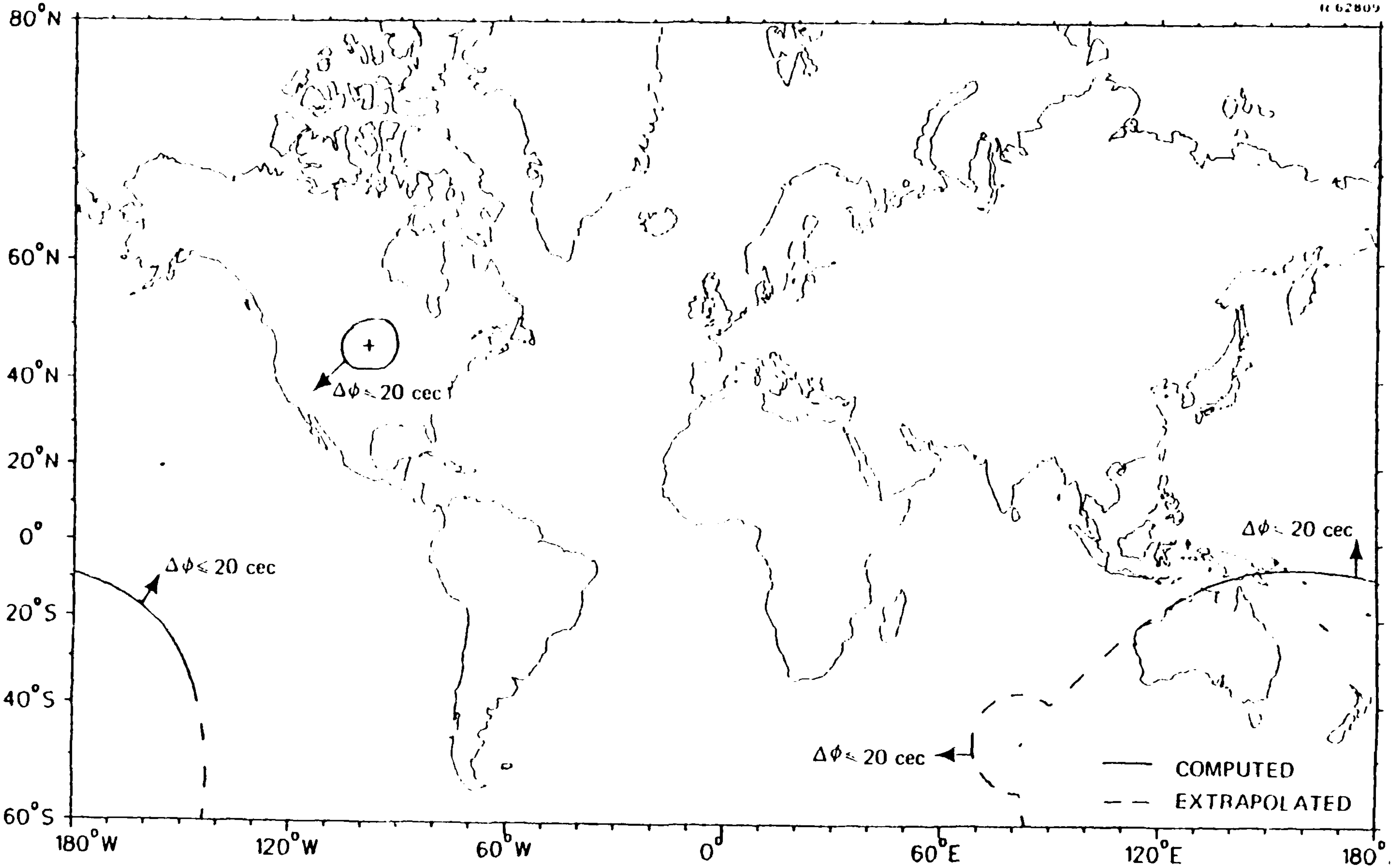


Figure 6. Lat-Long ONSOD Nighttime Modal Interference Model, La Reunion

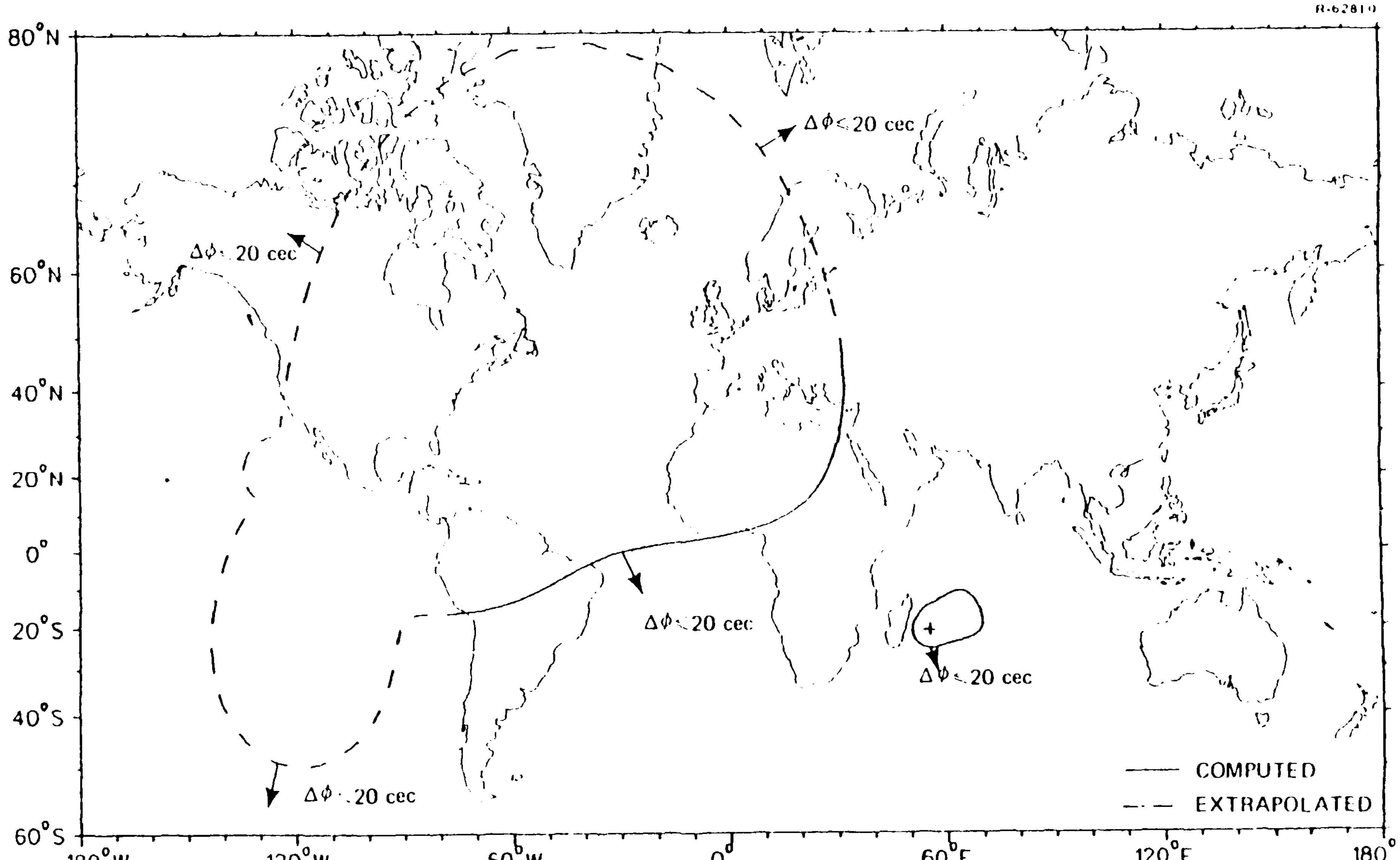


Figure 7. Lat-Long ONSOD Nighttime Modal Interference Model, Argentina

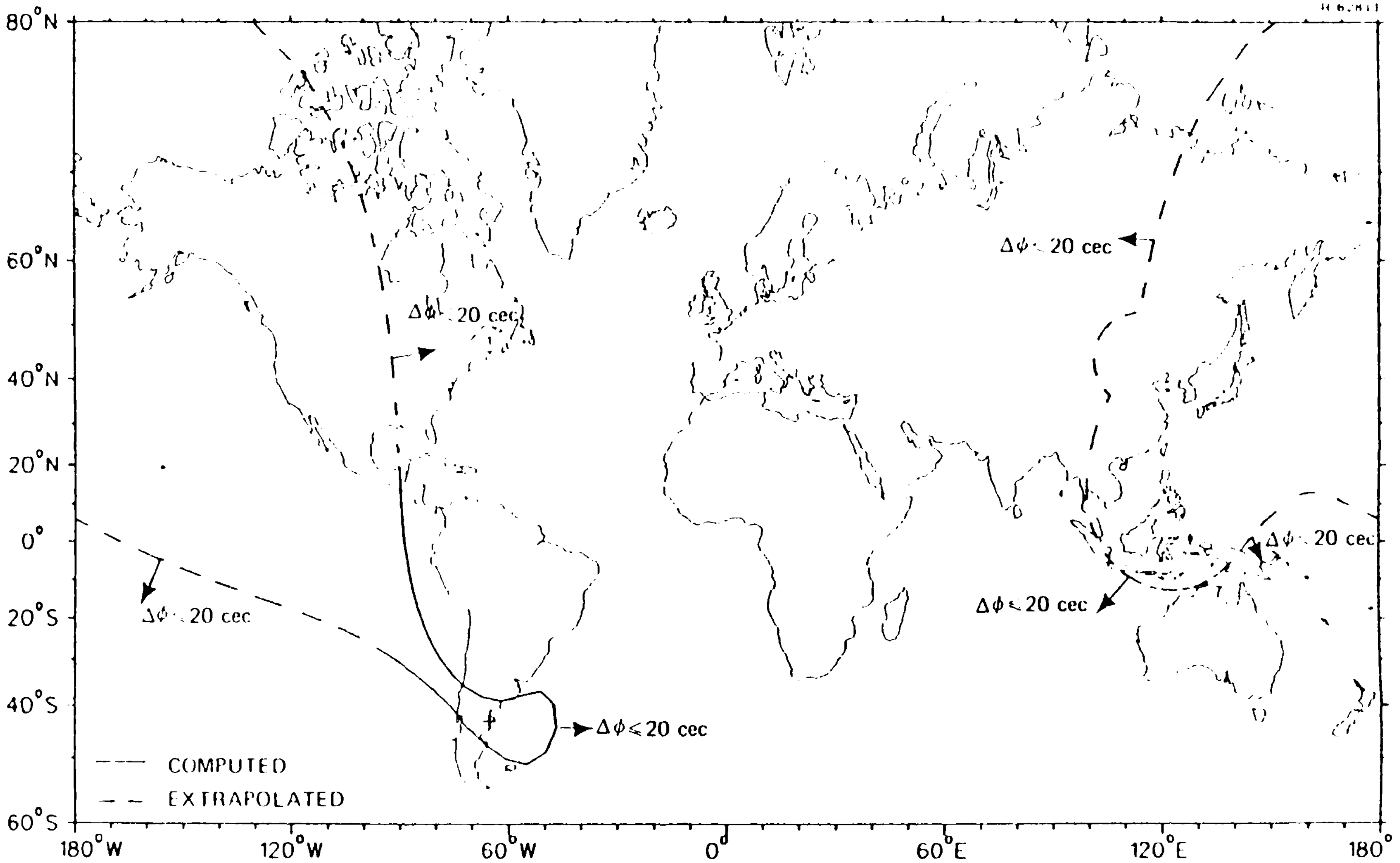


Figure 8. Lat-Long ONSOD Nighttime Modal Interference Model, Australia

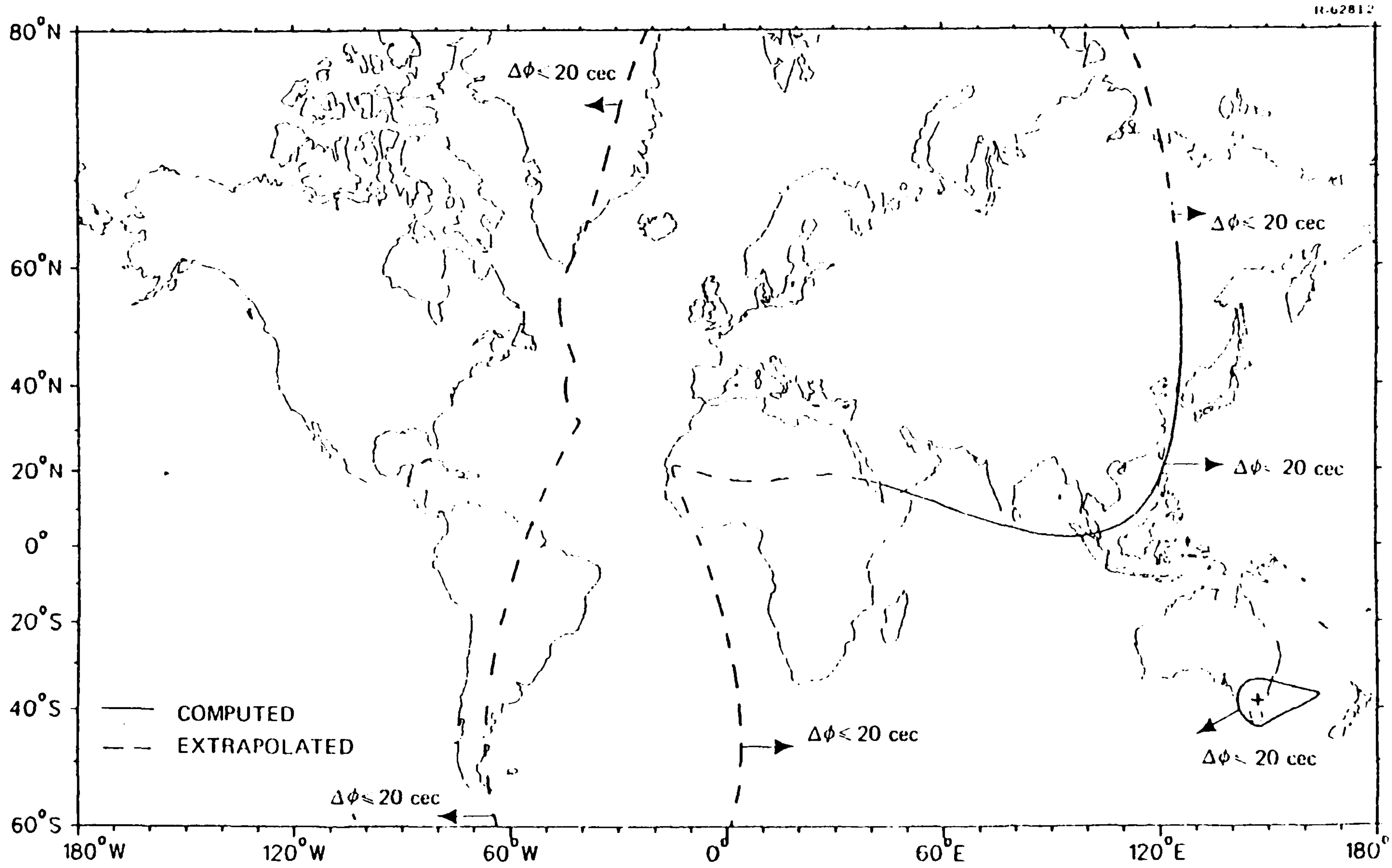


Figure 9. Lat-Long ONSOD Nighttime Modal Interference Model, Japan

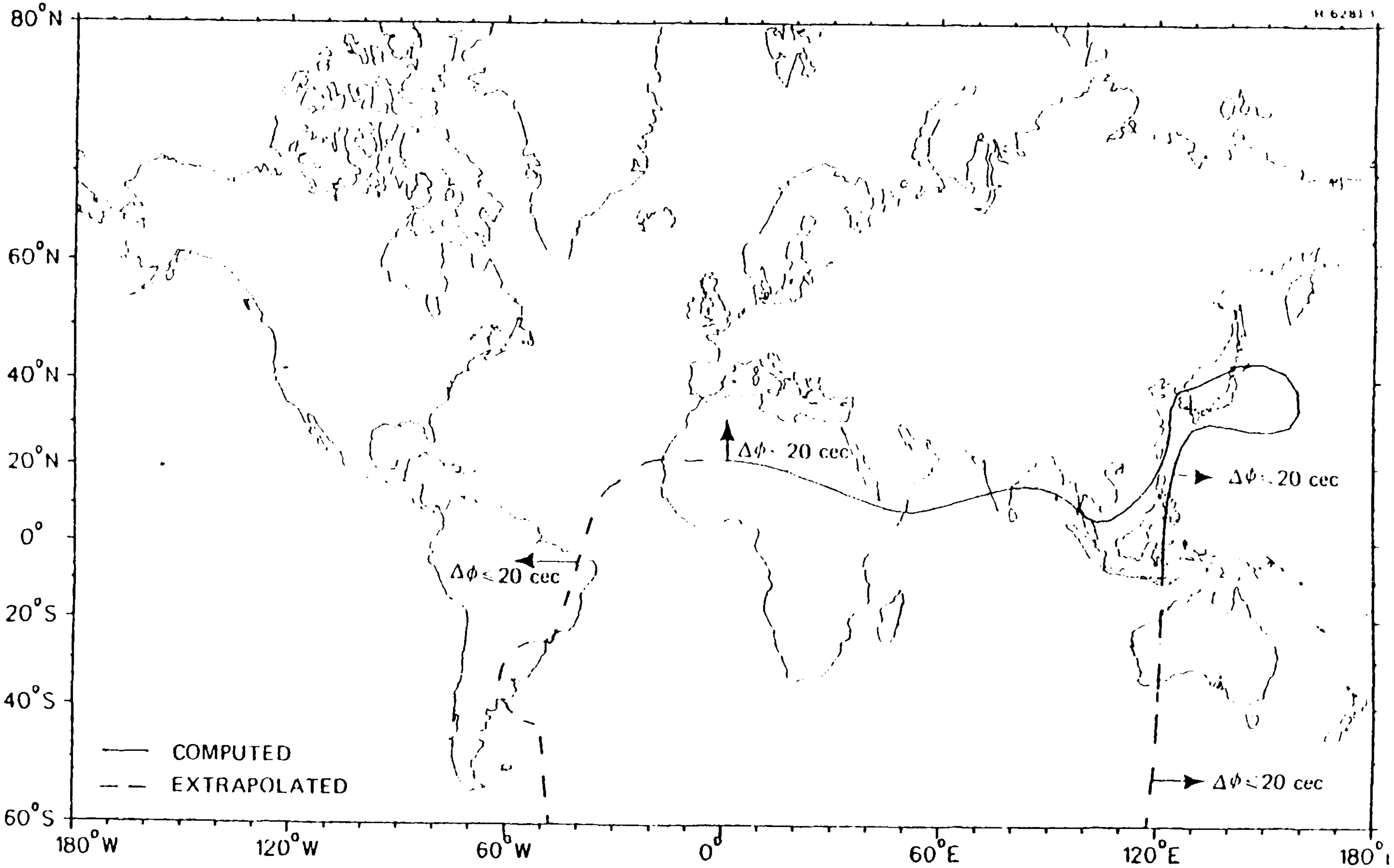


Figure 10.

6-JUL-87

NORWAY NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL

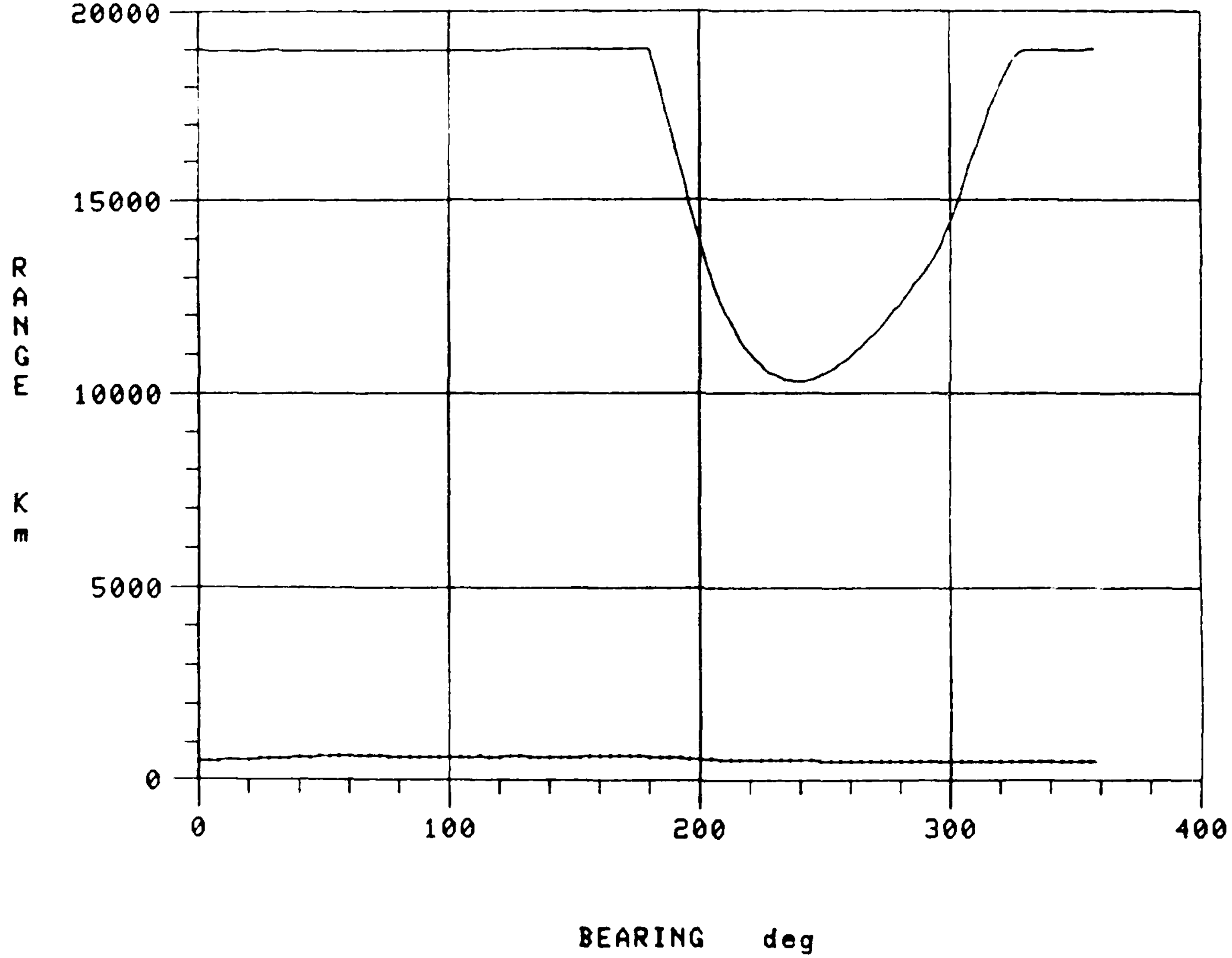


Figure 11.

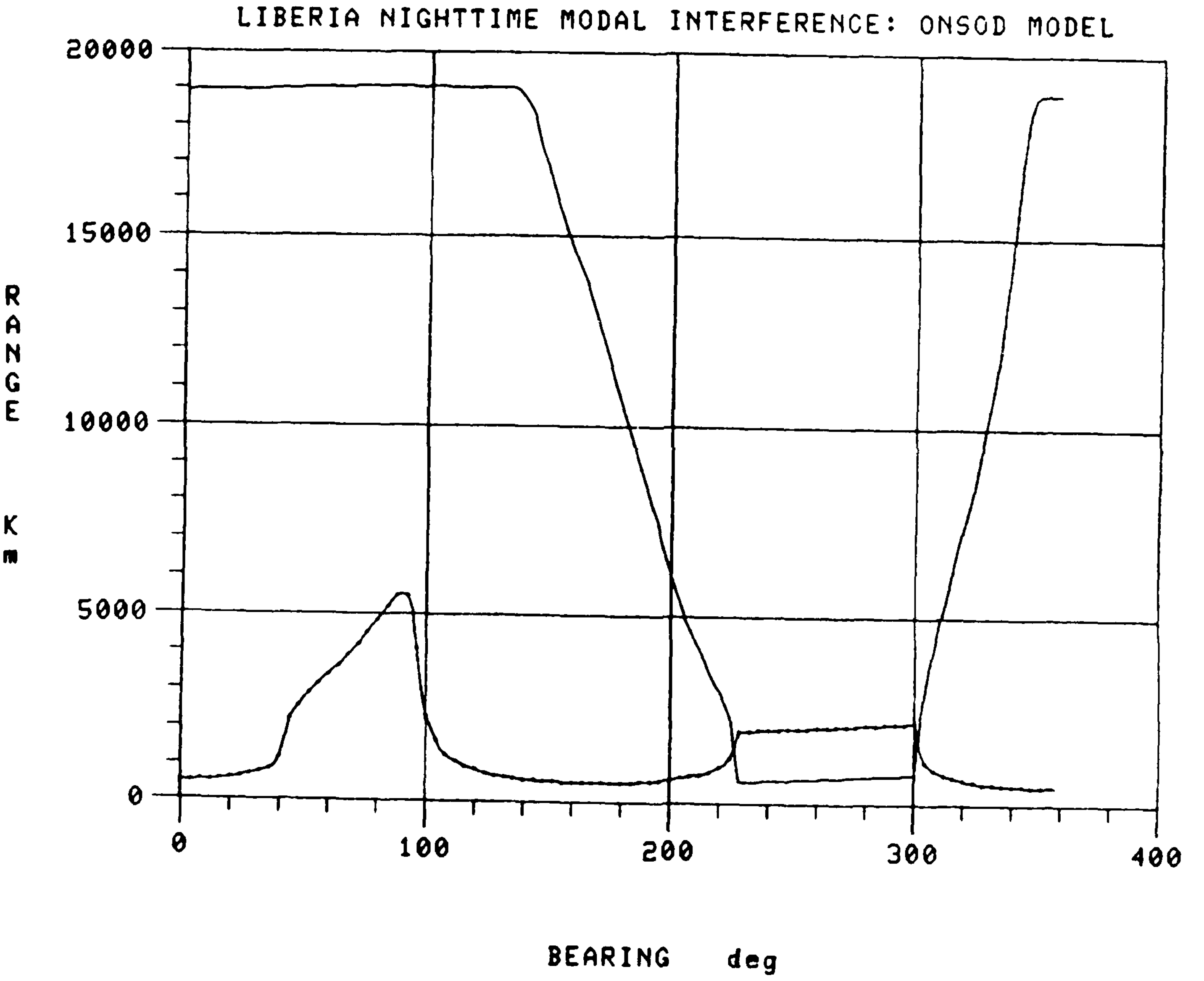

6-JUL-87
LIBERIA NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL
20000
15000
10000
5000
0
RANGE
Km
0
100
200
300
400
BEARING deg

Figure 12.

6-JUL-87

HAWAII NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL

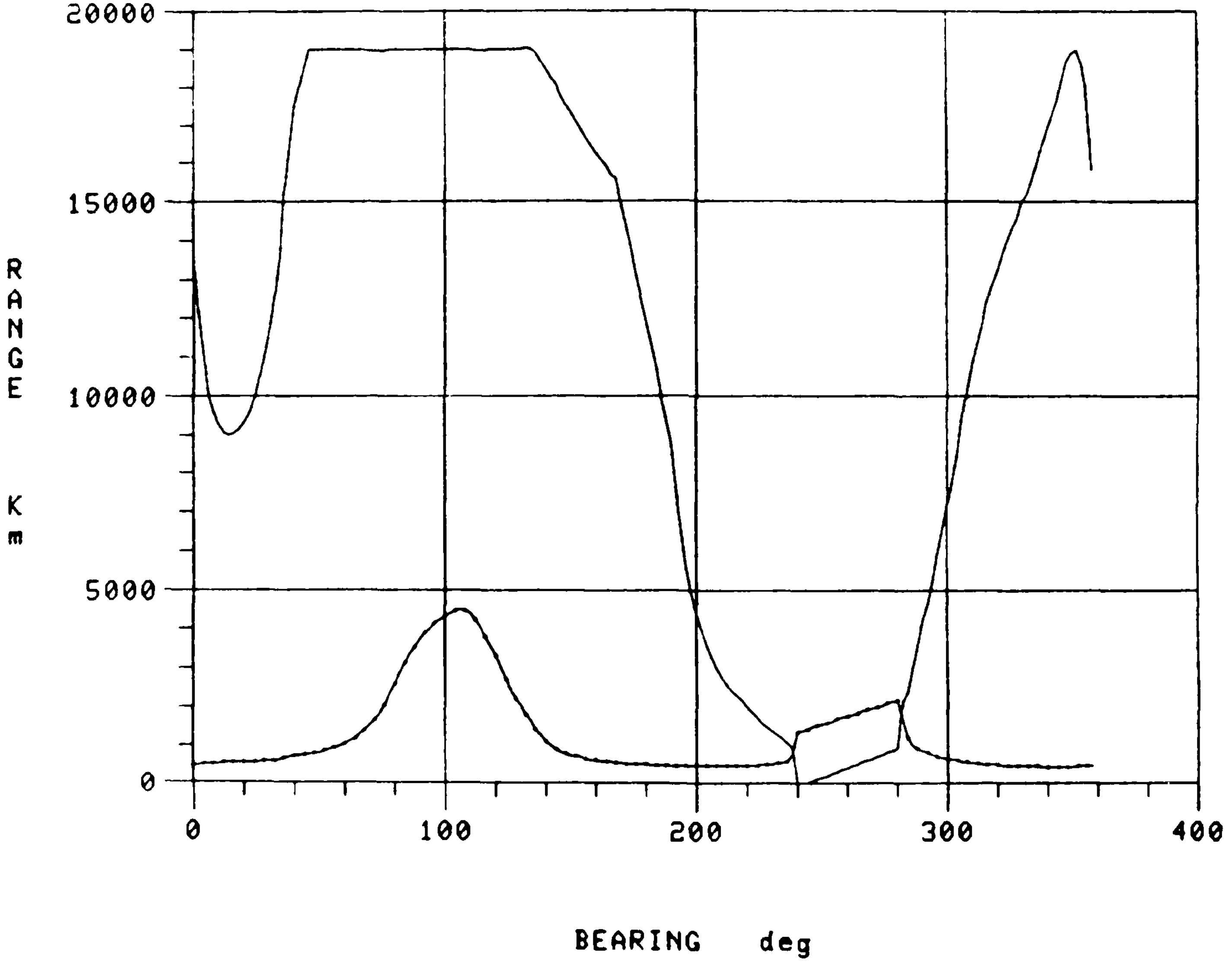


Figure 13.

6-JUL-87

NORTH DAKOTA NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL

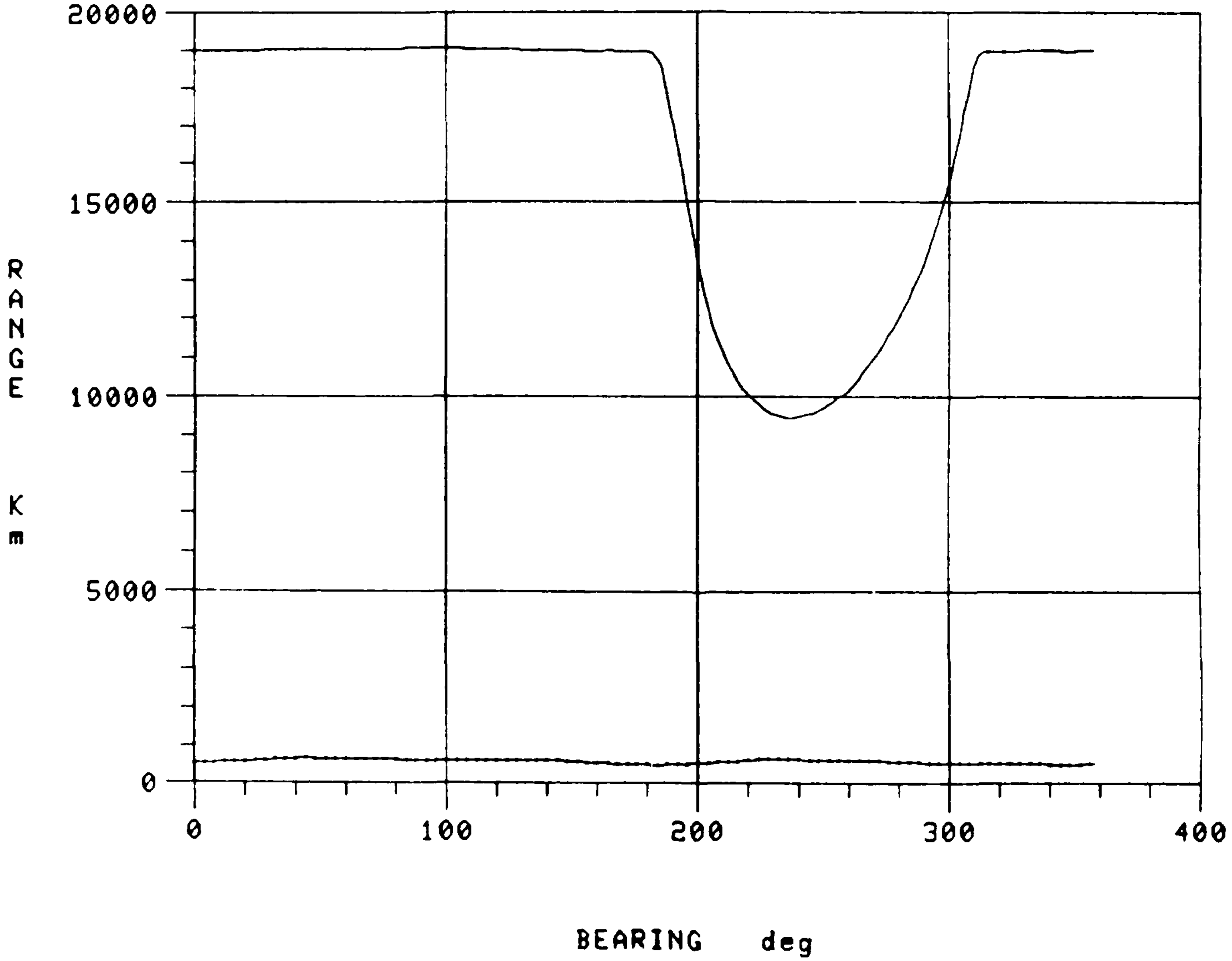


Figure 14. 6-JUL-87

LA_REUNION NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL

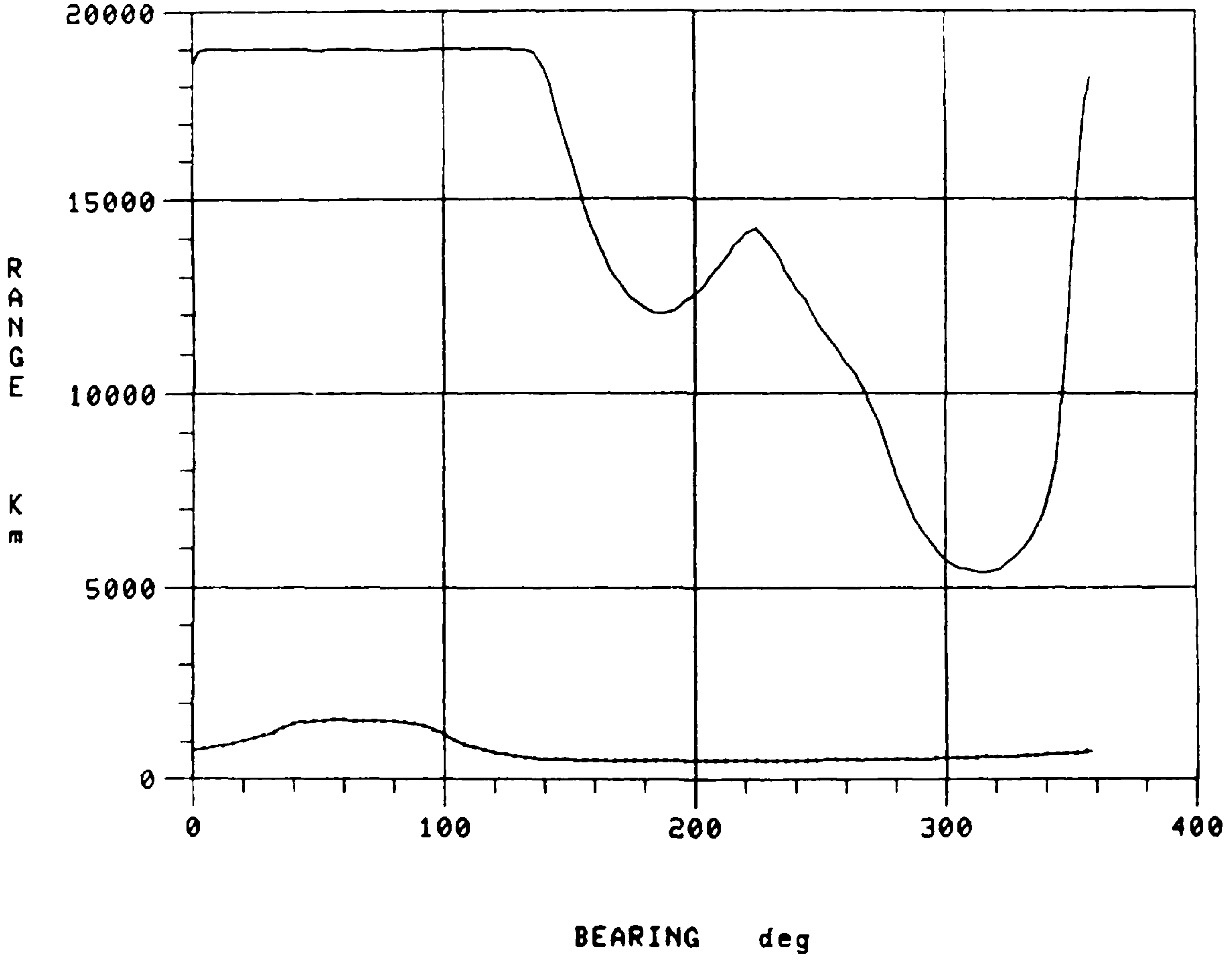


Figure 15.

21-SEP-87

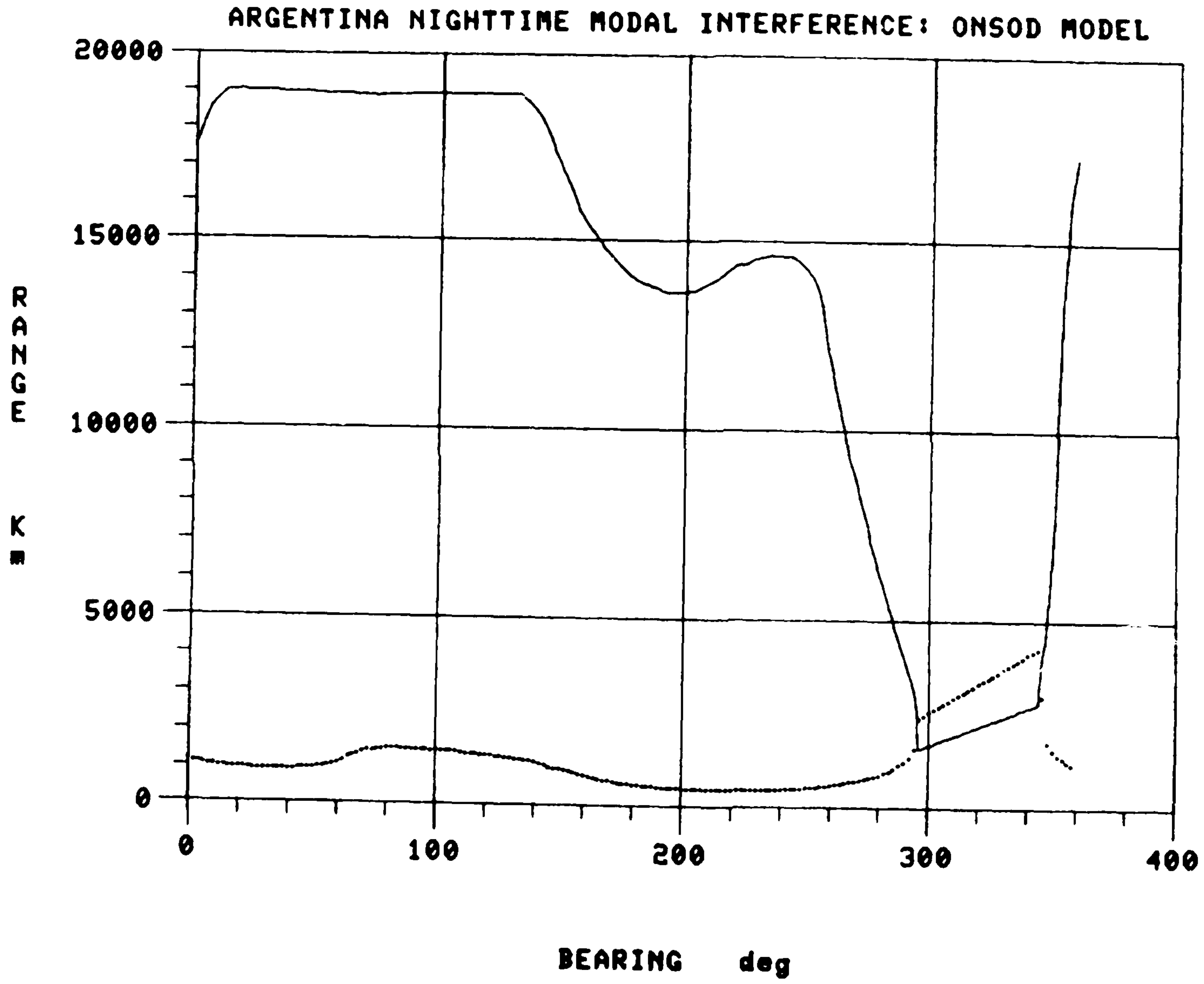


Figure 16.

6-JUL-87

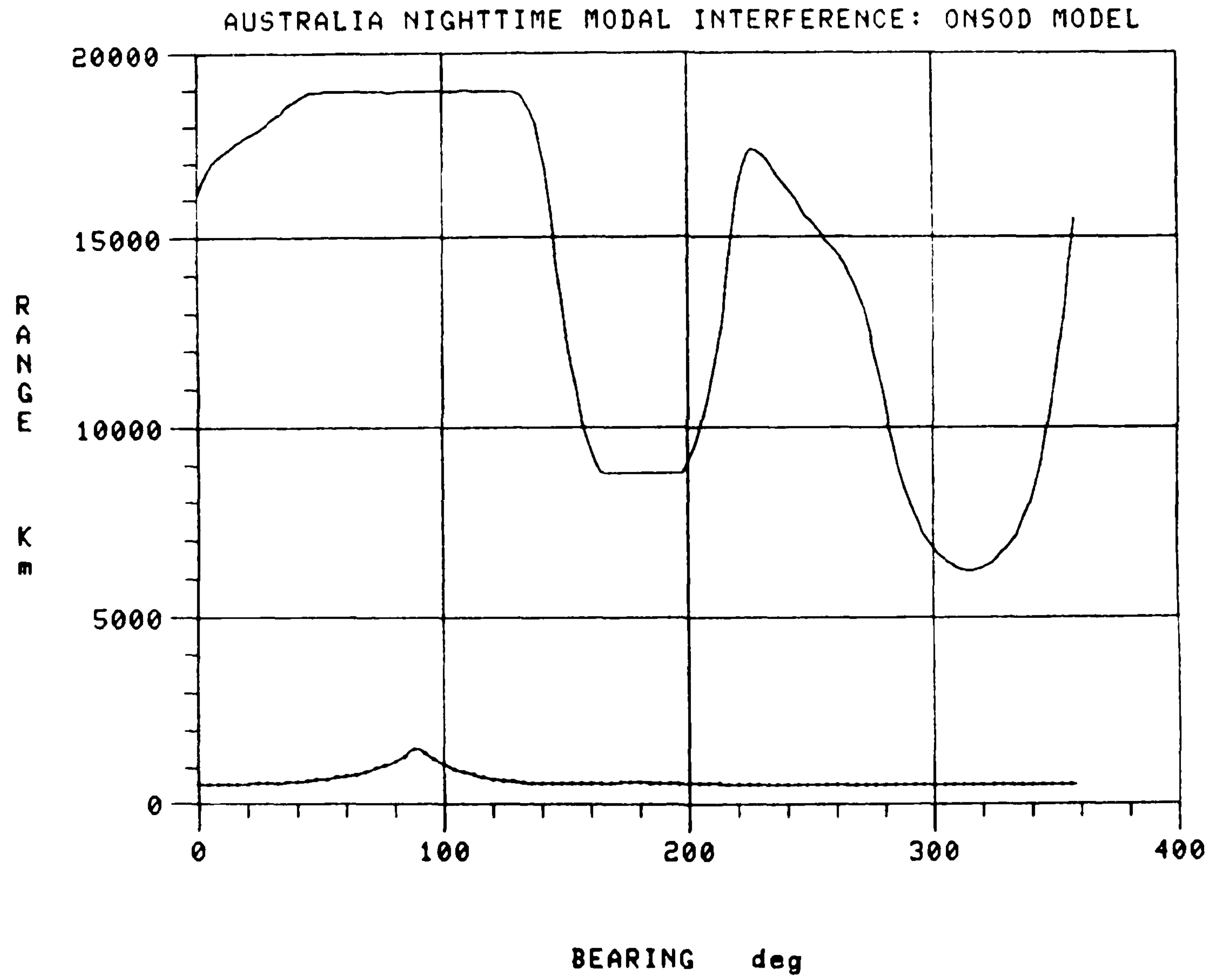


Figure 17. 6-JUL-87

JAPAN NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL

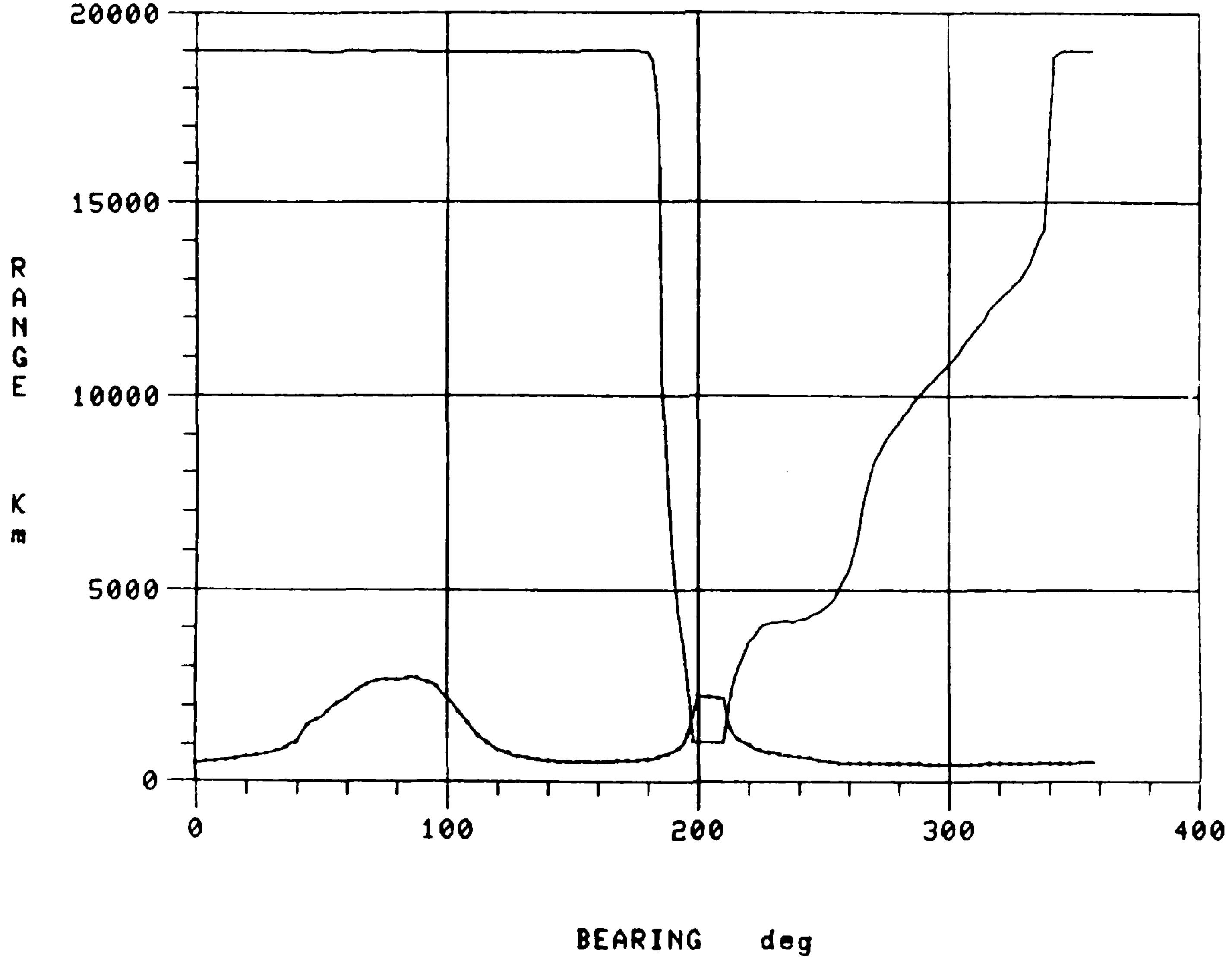


Figure 18. 6-JUL-87

NORWAY NIGHTTIME MODAL INTERFERENCE: DREC MODEL

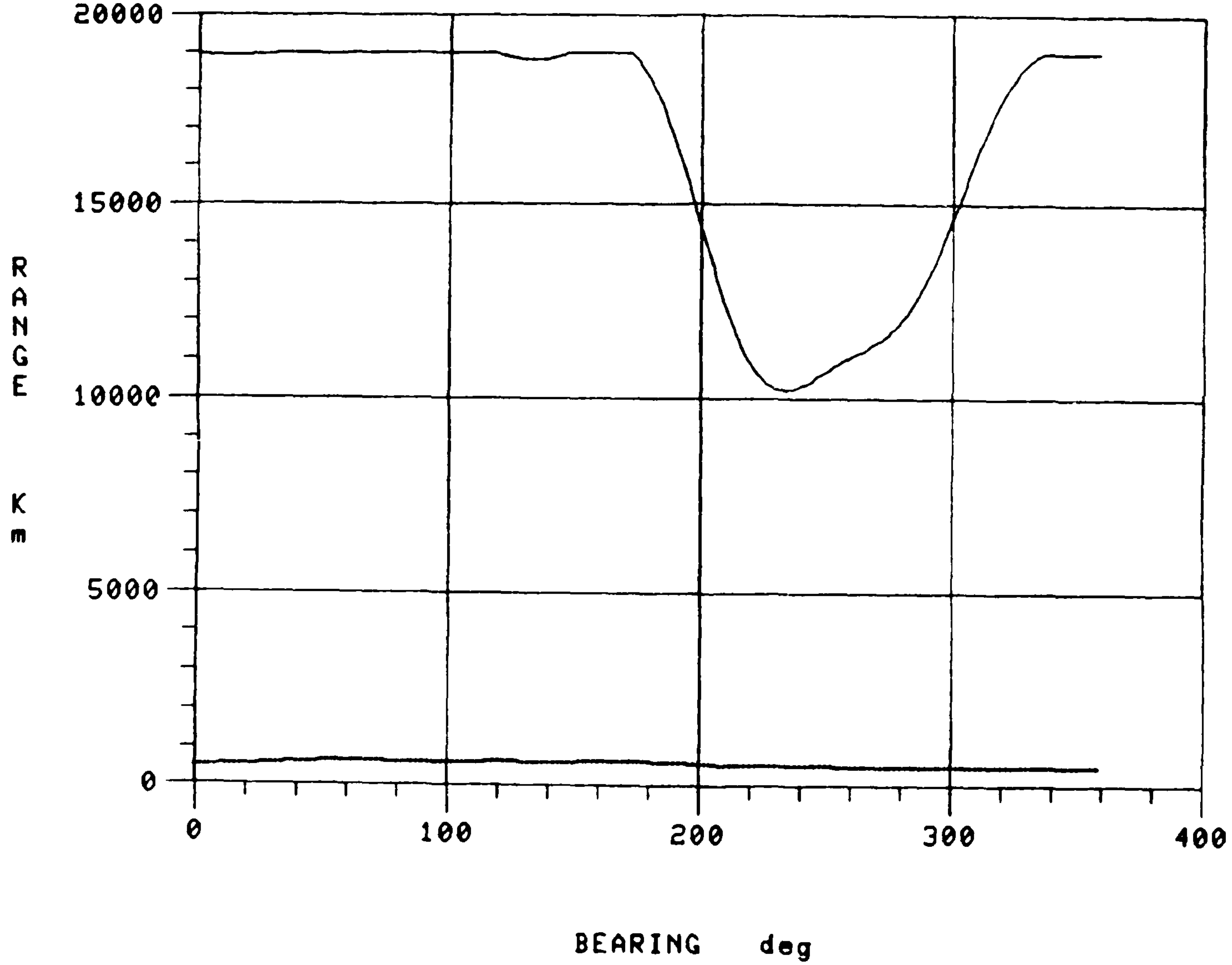


Figure 19.

6-JUL-87

LIBERIA NIGHTTIME MODAL INTERFERENCE: DRED MODEL

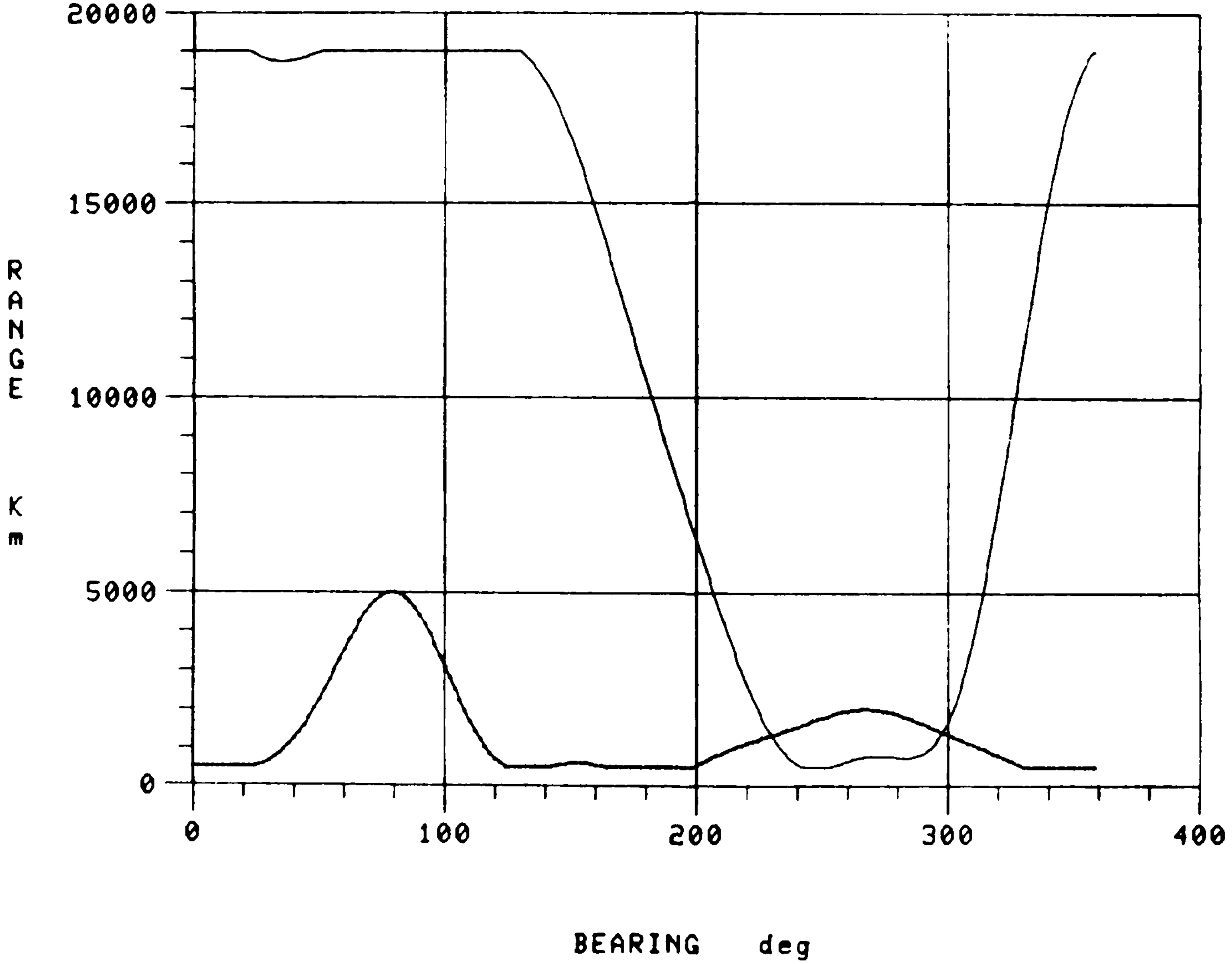


Figure 20.

6-JUL-87

HAWAII NIGHTTIME MODAL INTERFERENCE: DREO MODEL

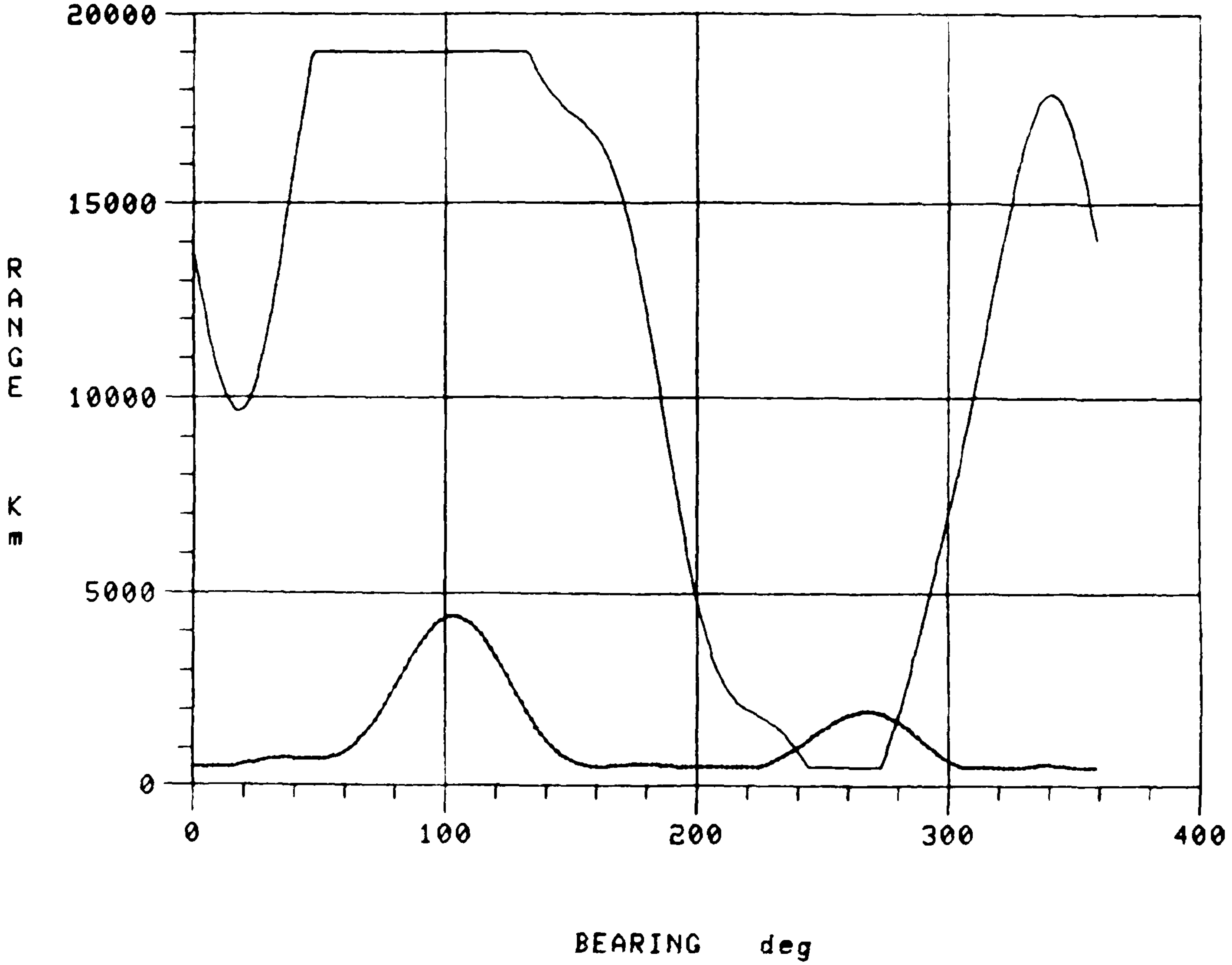


Figure 21.

6-JUL-87

NORTH DAKOTA NIGHTTIME MODAL INTERFERENCE: DREO MODEL

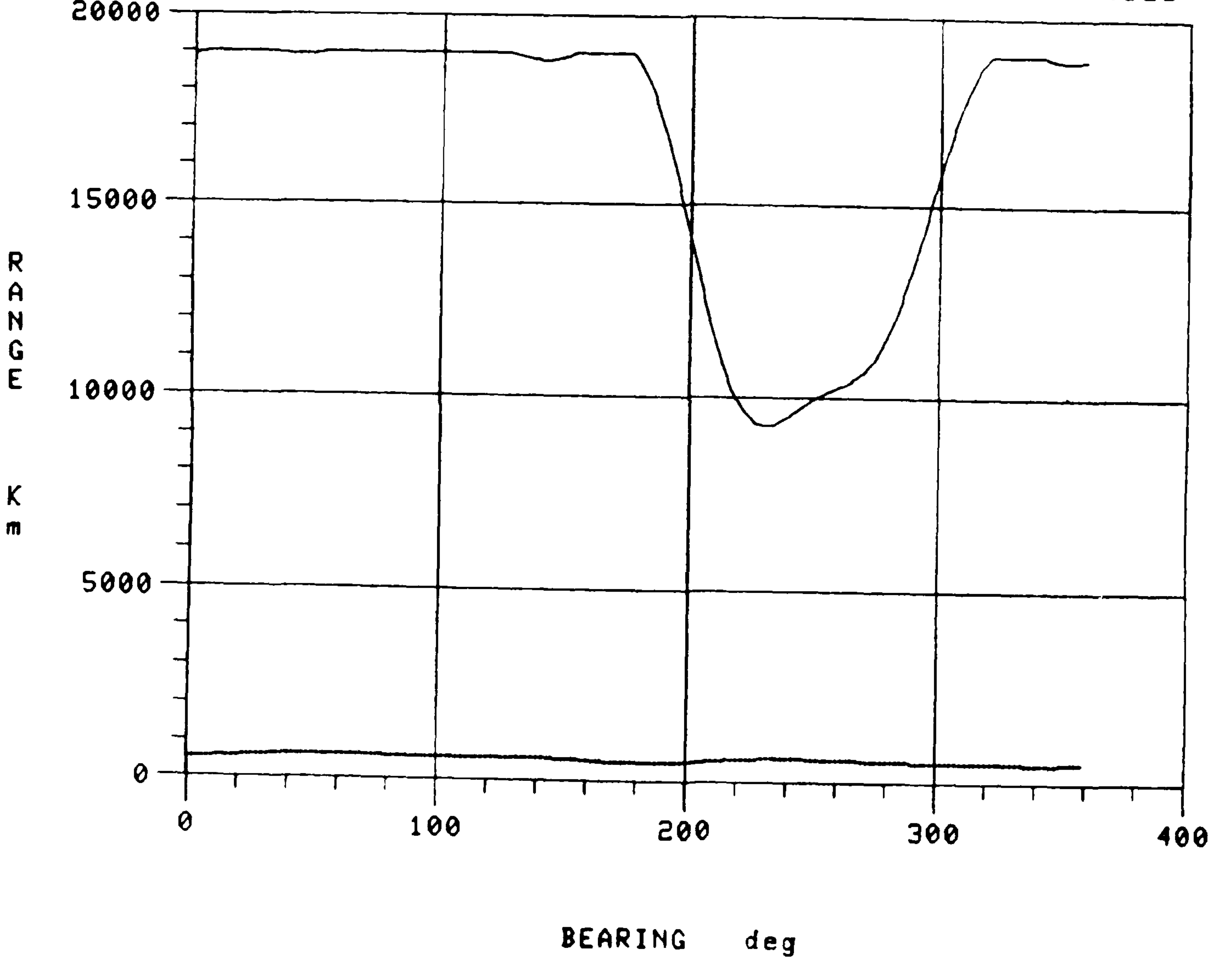


Figure 22.

6-JUL-87

LA_REUNION NIGHTTIME MODAL INTERFERENCE: DREO MODEL

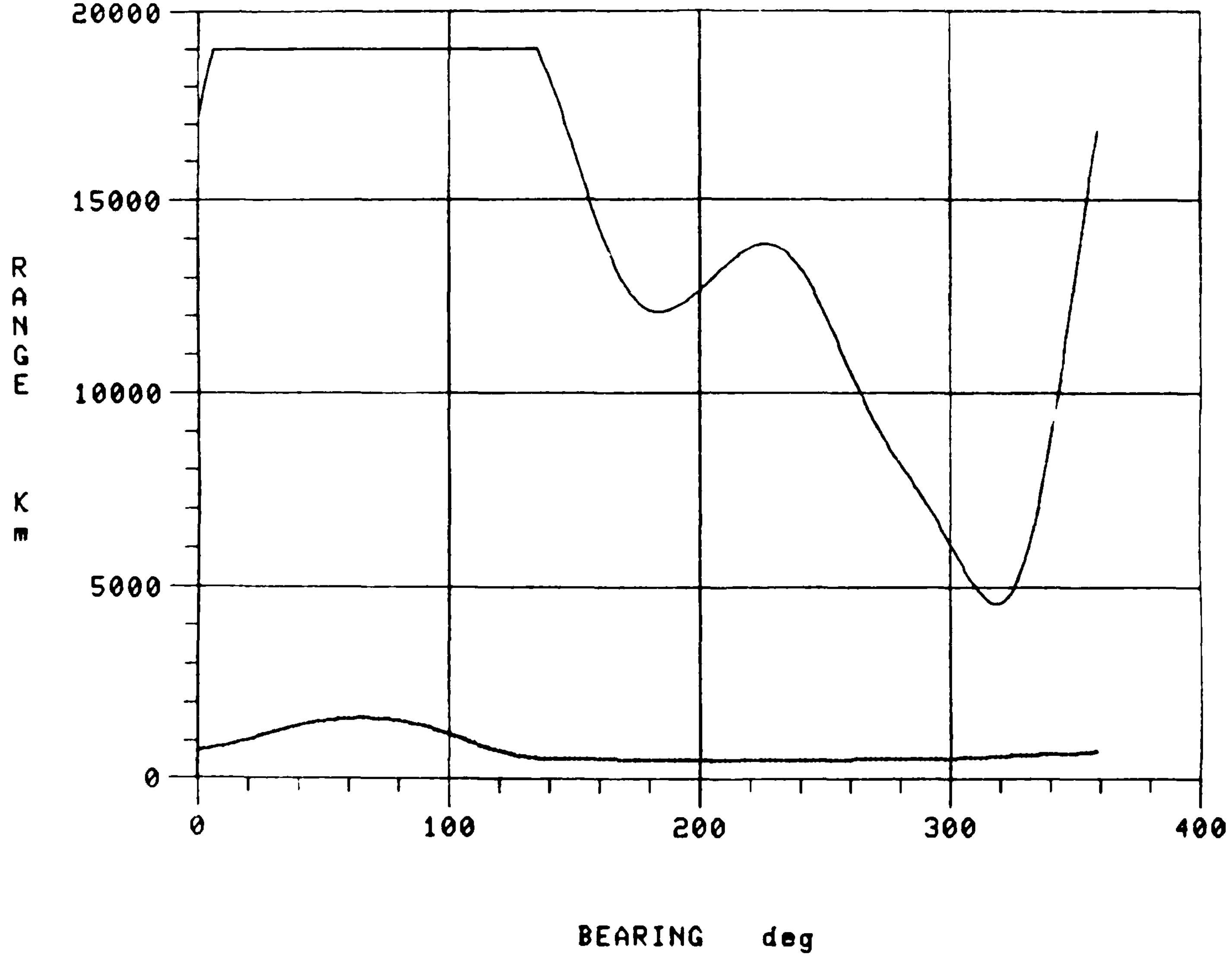


Figure 23.

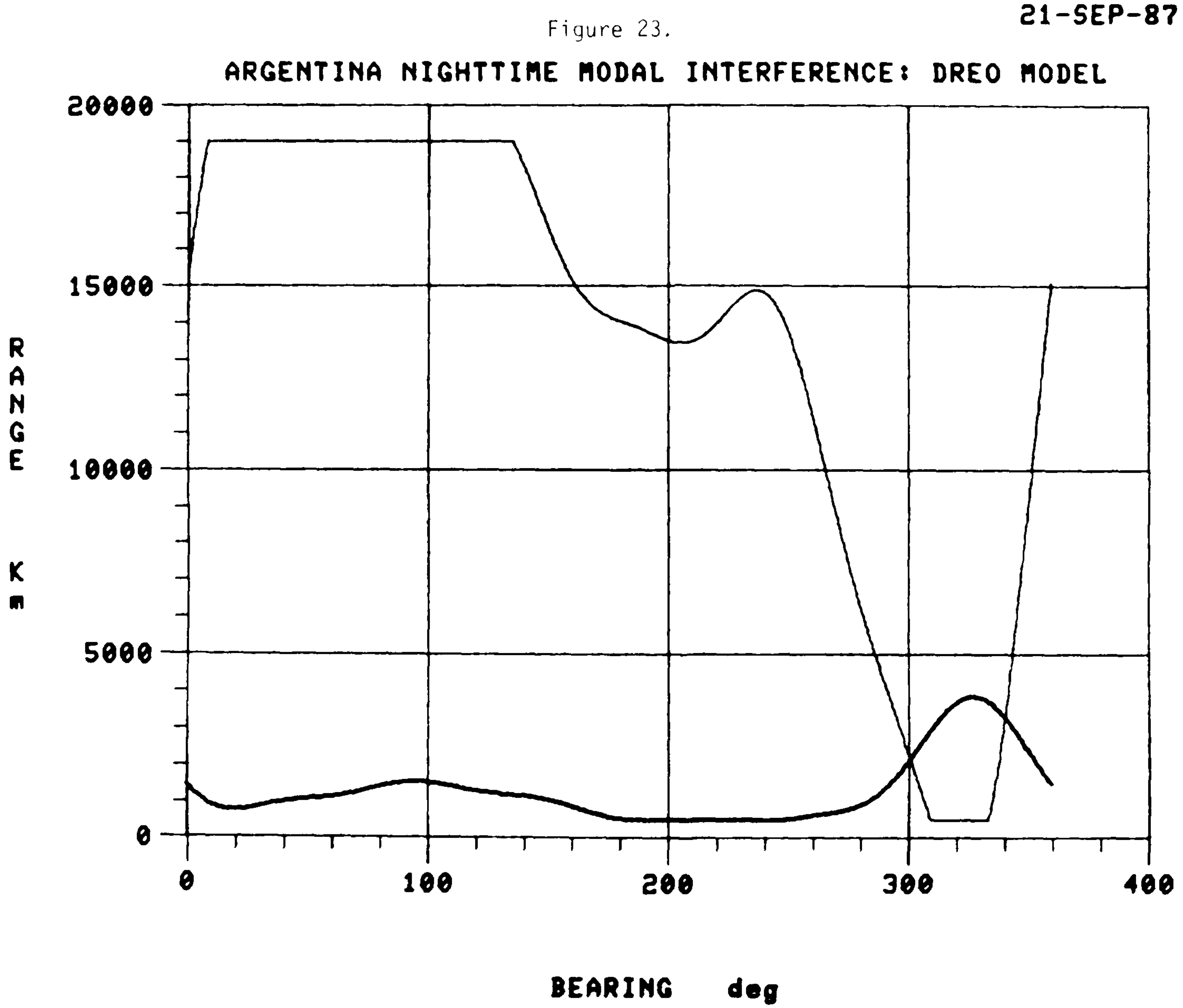

21-SEP-87
ARGENTINA NIGHTTIME MODAL INTERFERENCE: DREO MODEL
20000
15000
10000
5000
0
RANGE Km
0
100
200
300
400
BEARING deg

Figure 24.

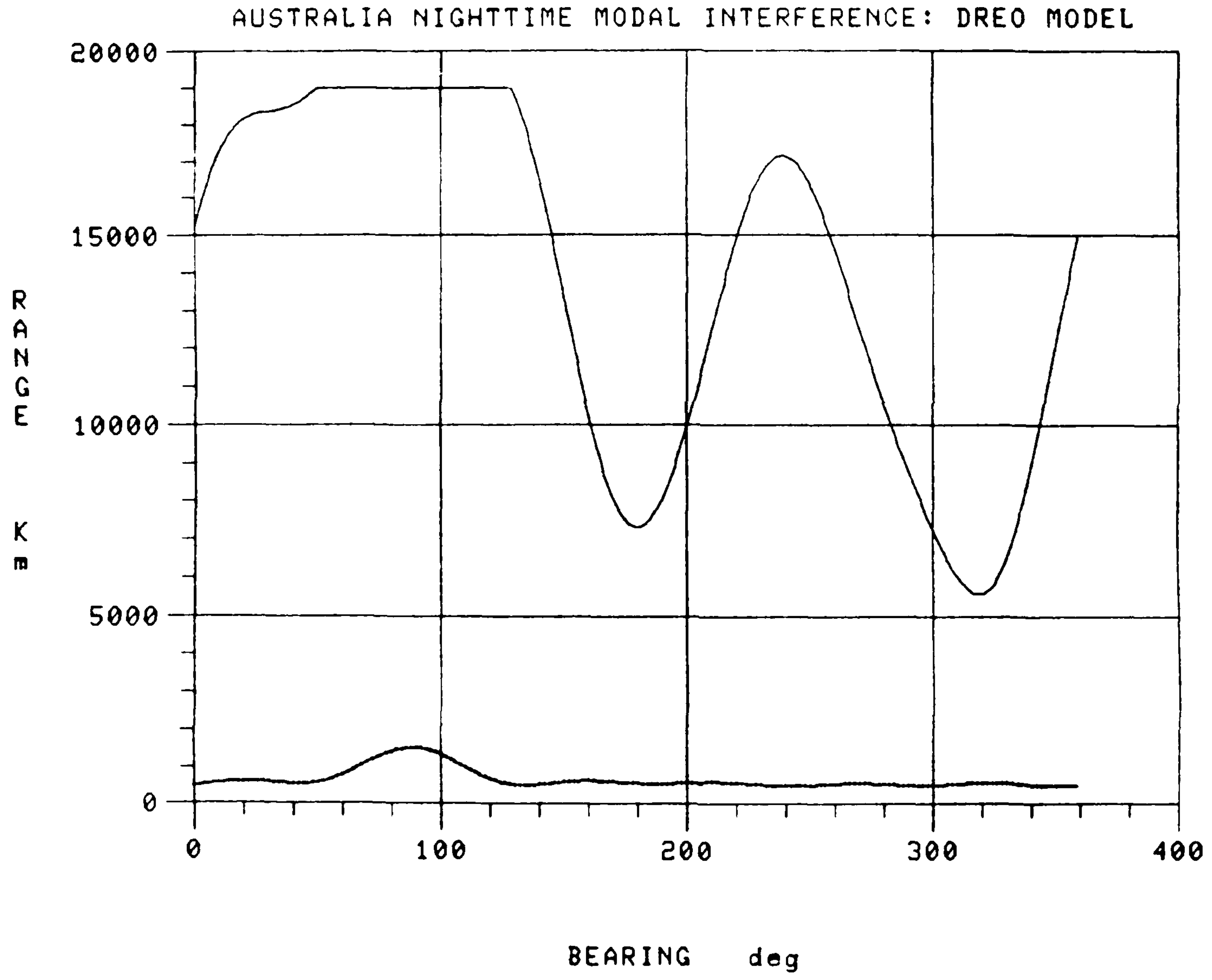

6-JUL-87
AUSTRALIA NIGHTTIME MODAL INTERFERENCE: DREO MODEL
RANGE Km
20000
15000
10000
5000
0
0
100
200
300
400
BEARING deg

Figure 25.

6-JUL-87

JAPAN NIGHTTIME MODAL INTERFERENCE: DREO MODEL

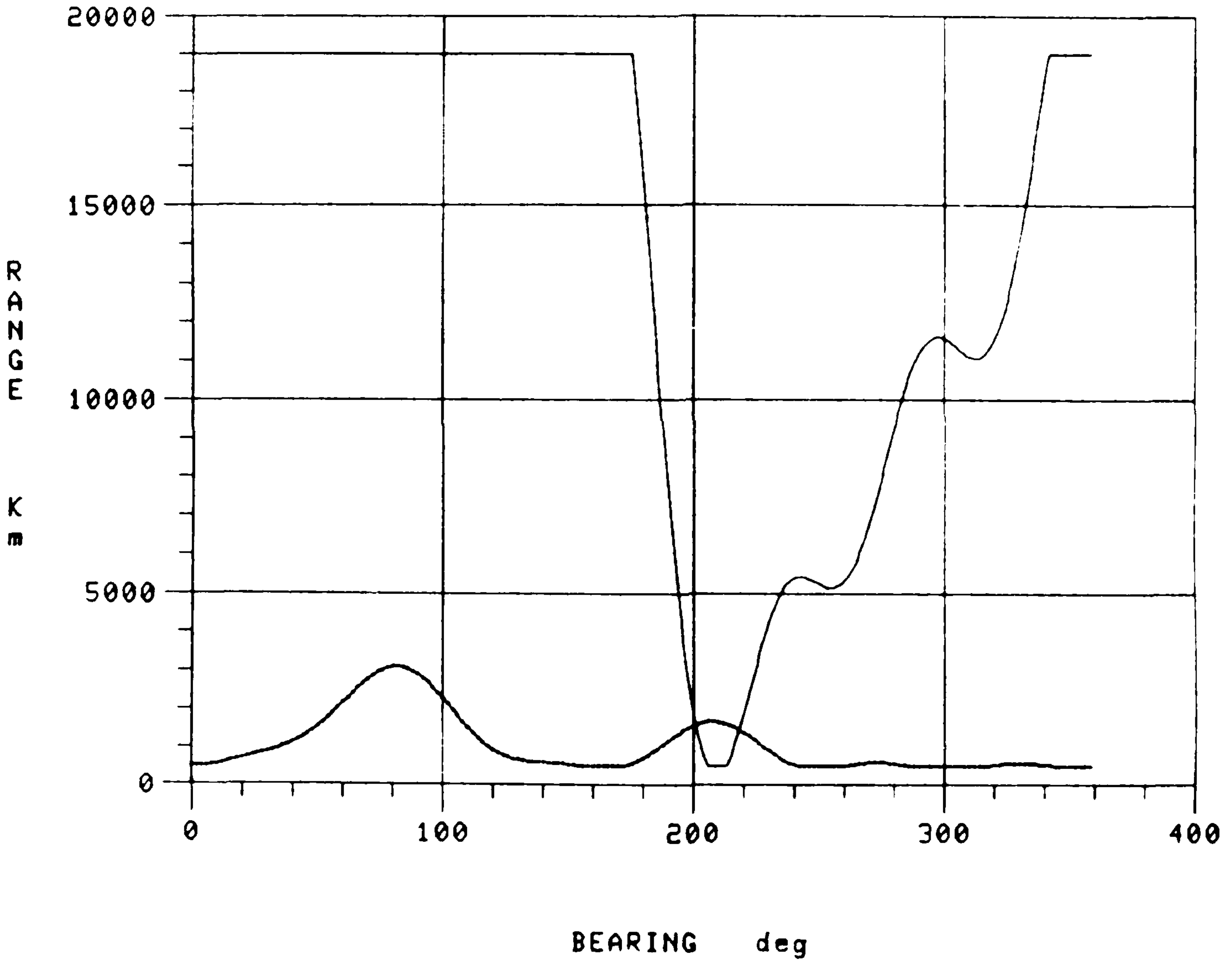


Figure 26.

6-JUL-87

NORWAY NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL

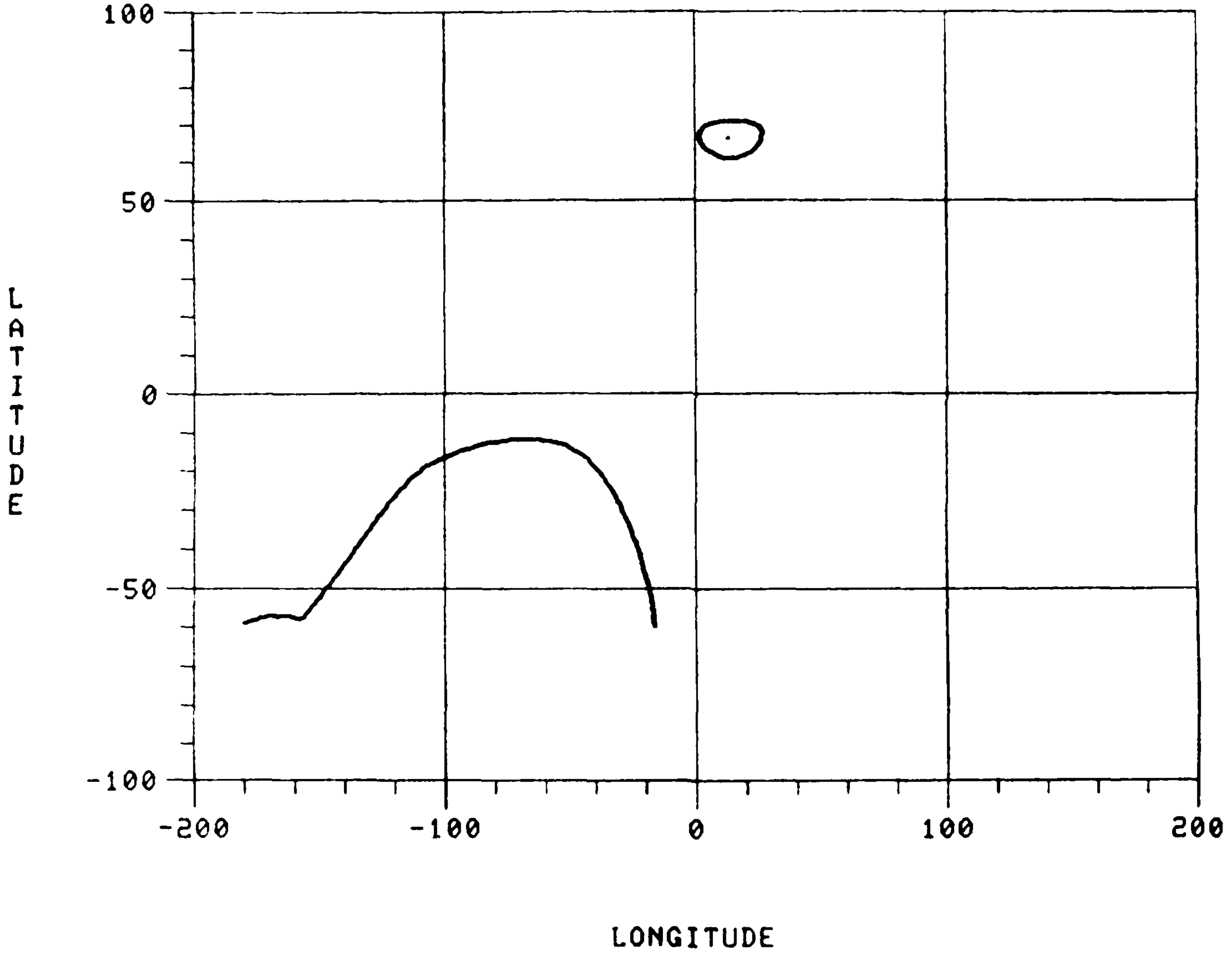


Figure 27.

6-JUL-87

NORWAY NIGHTTIME MODAL INTERFERENCE: DREO MODEL

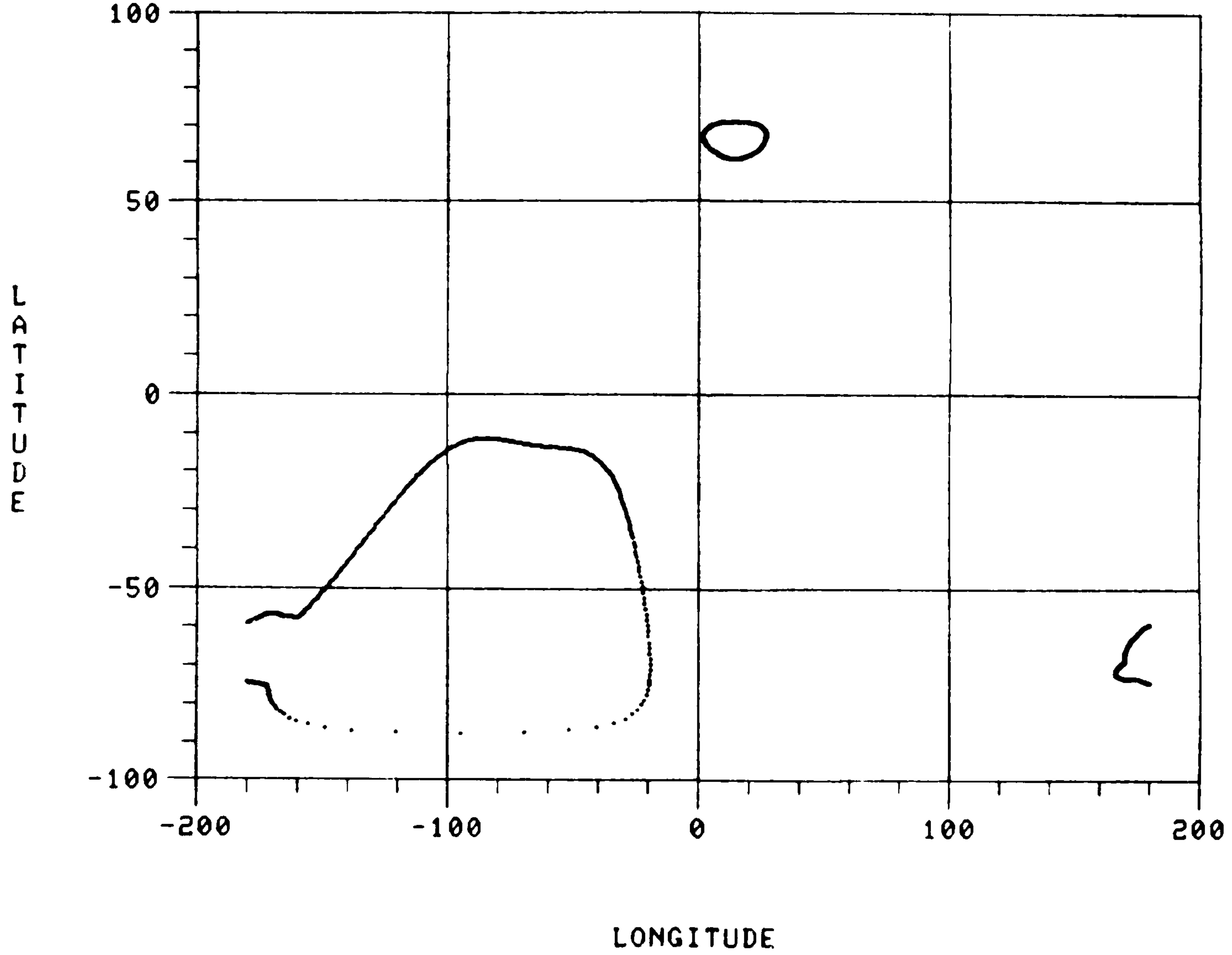


Figure 28.

6-JUL-87

LIBERIA NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL

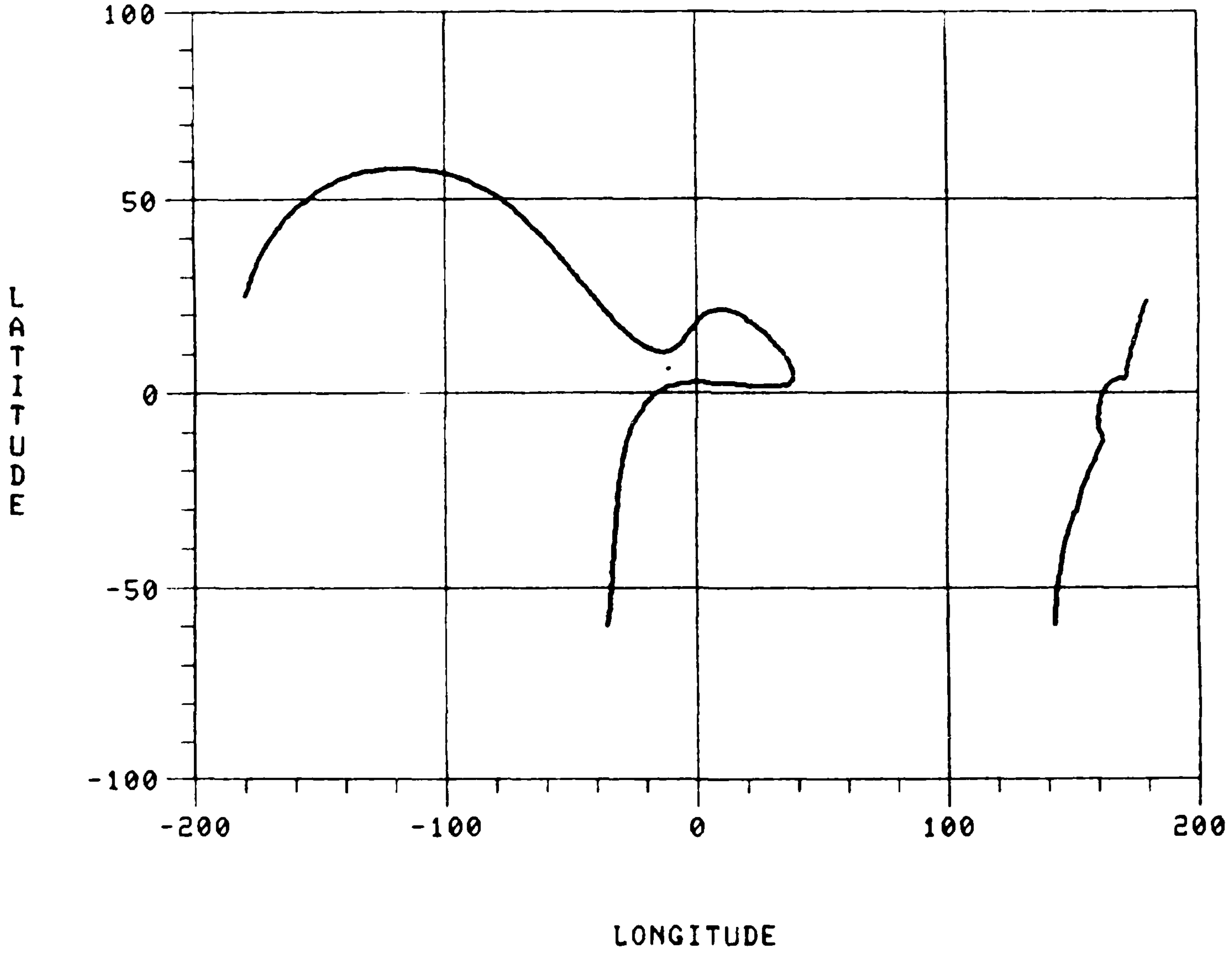


Figure 29.

6-JUL-87

LIBERIA NIGHTTIME MODAL INTERFERENCE: DREO MODEL

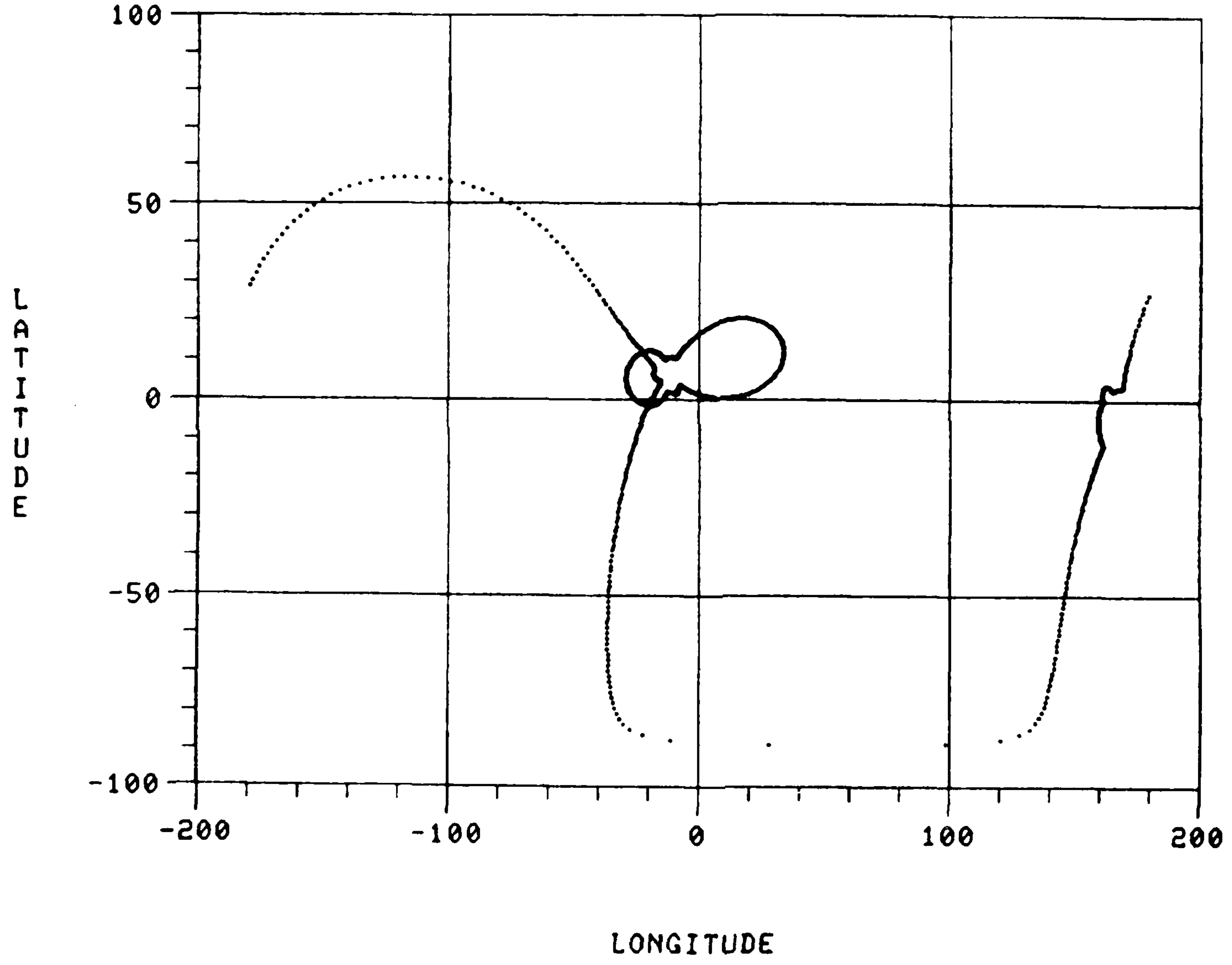


Figure 30.

6-JUL-87

HAWAII NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL

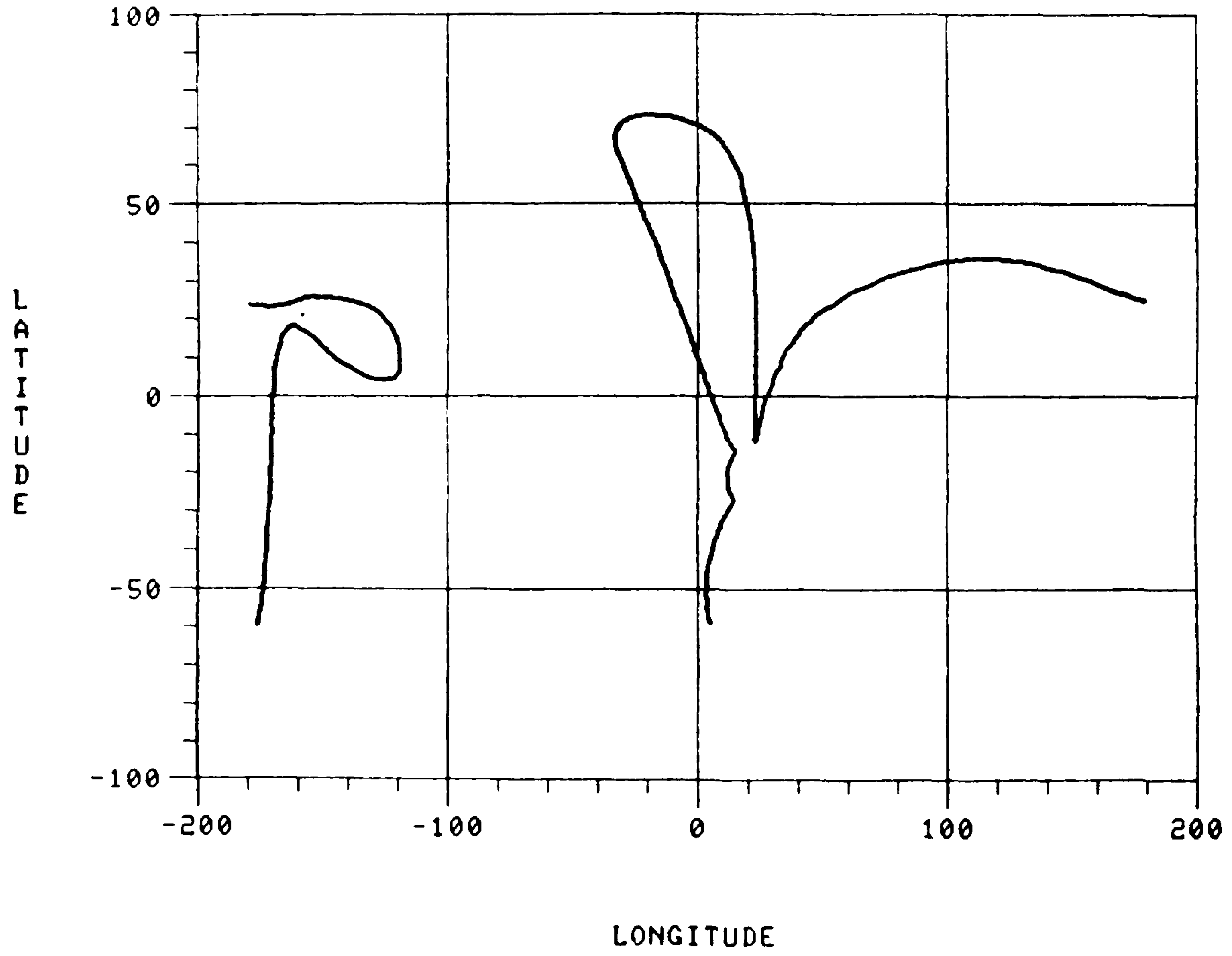


Figure 31. 6-JUL-87

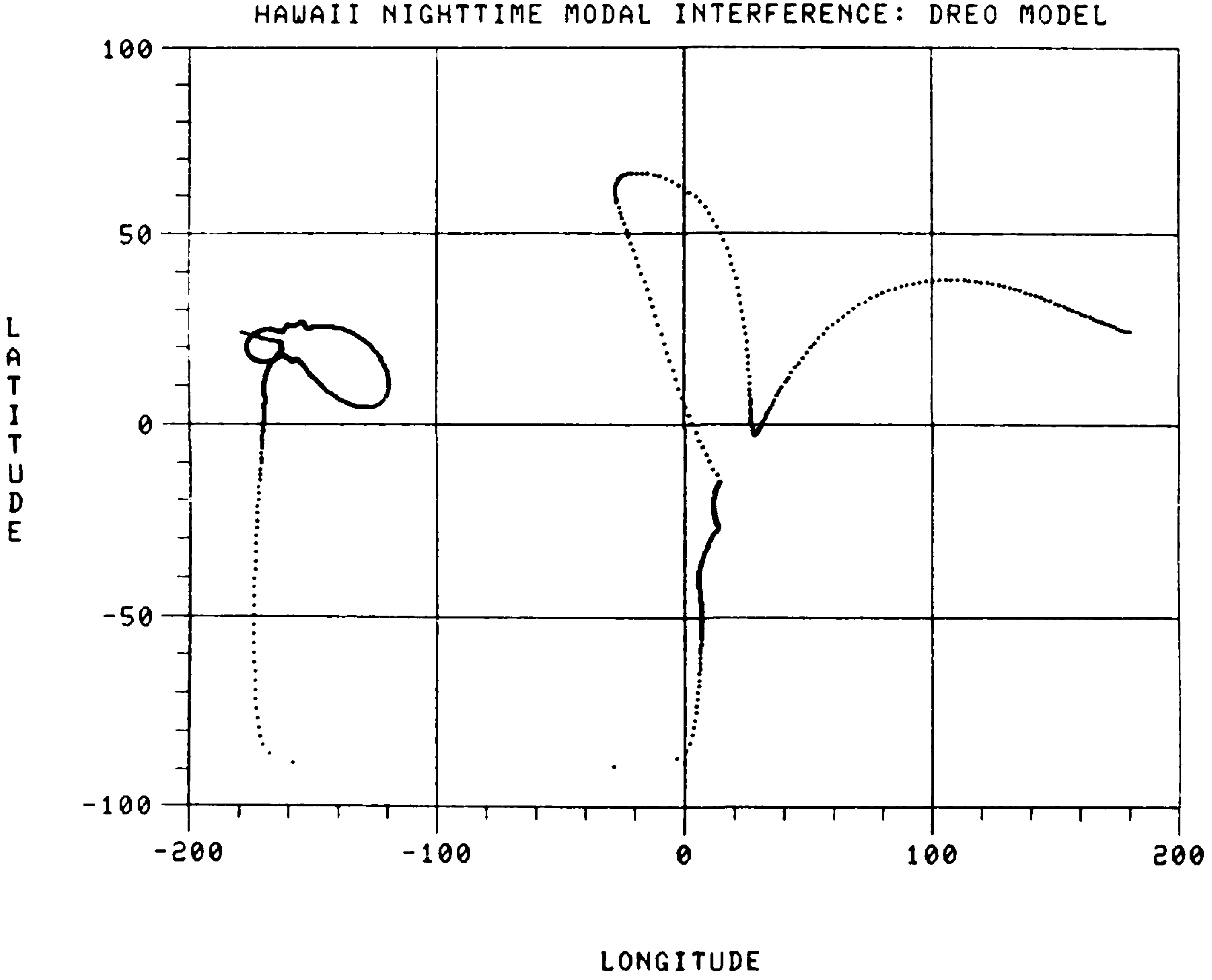


Figure 32.

6-JUL-87

NORTH DAKOTA NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL

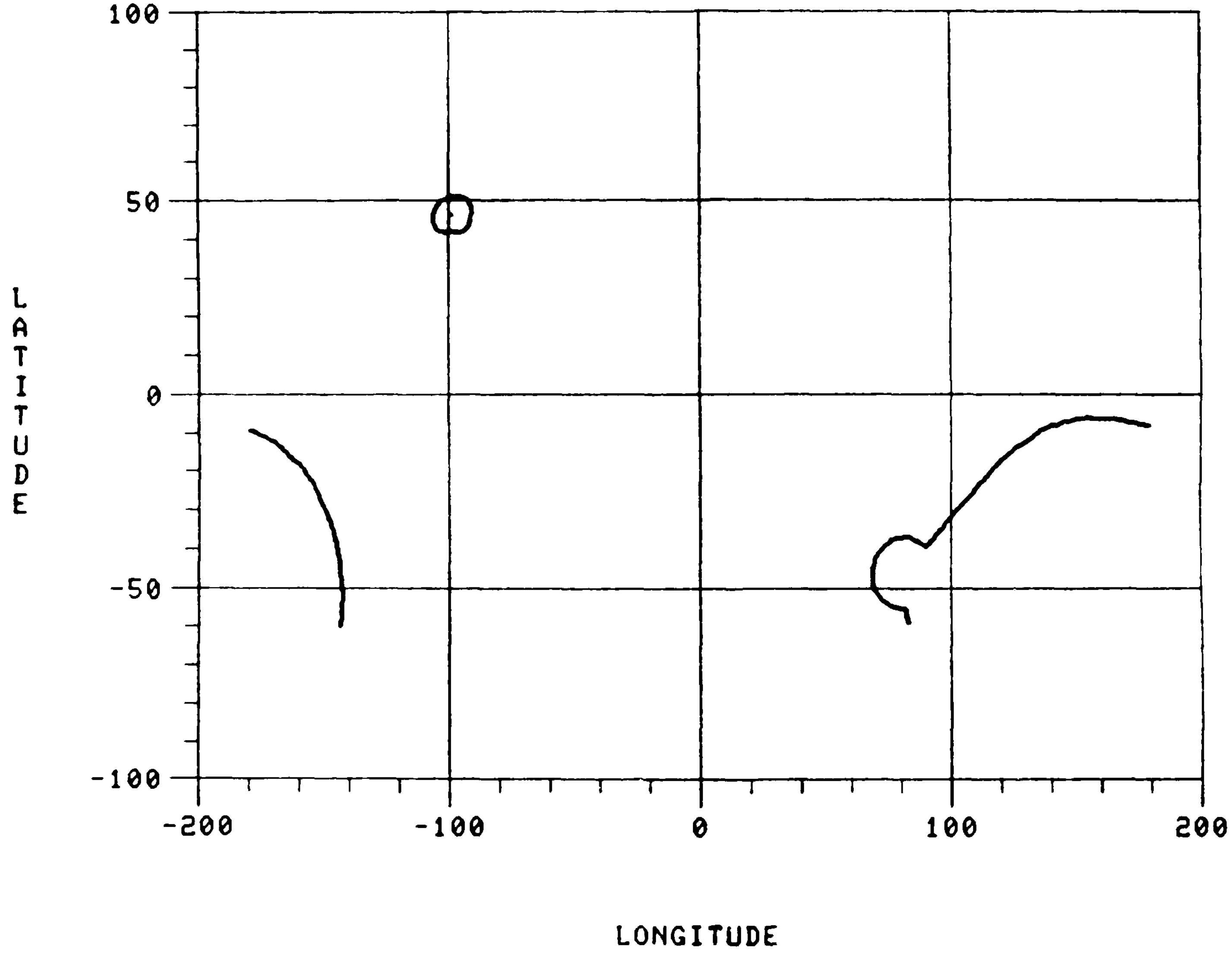


Figure 33.

6-JUL-87

NORTH_DAKOTA NIGHTTIME MODAL INTERFERENCE: DREO MODEL

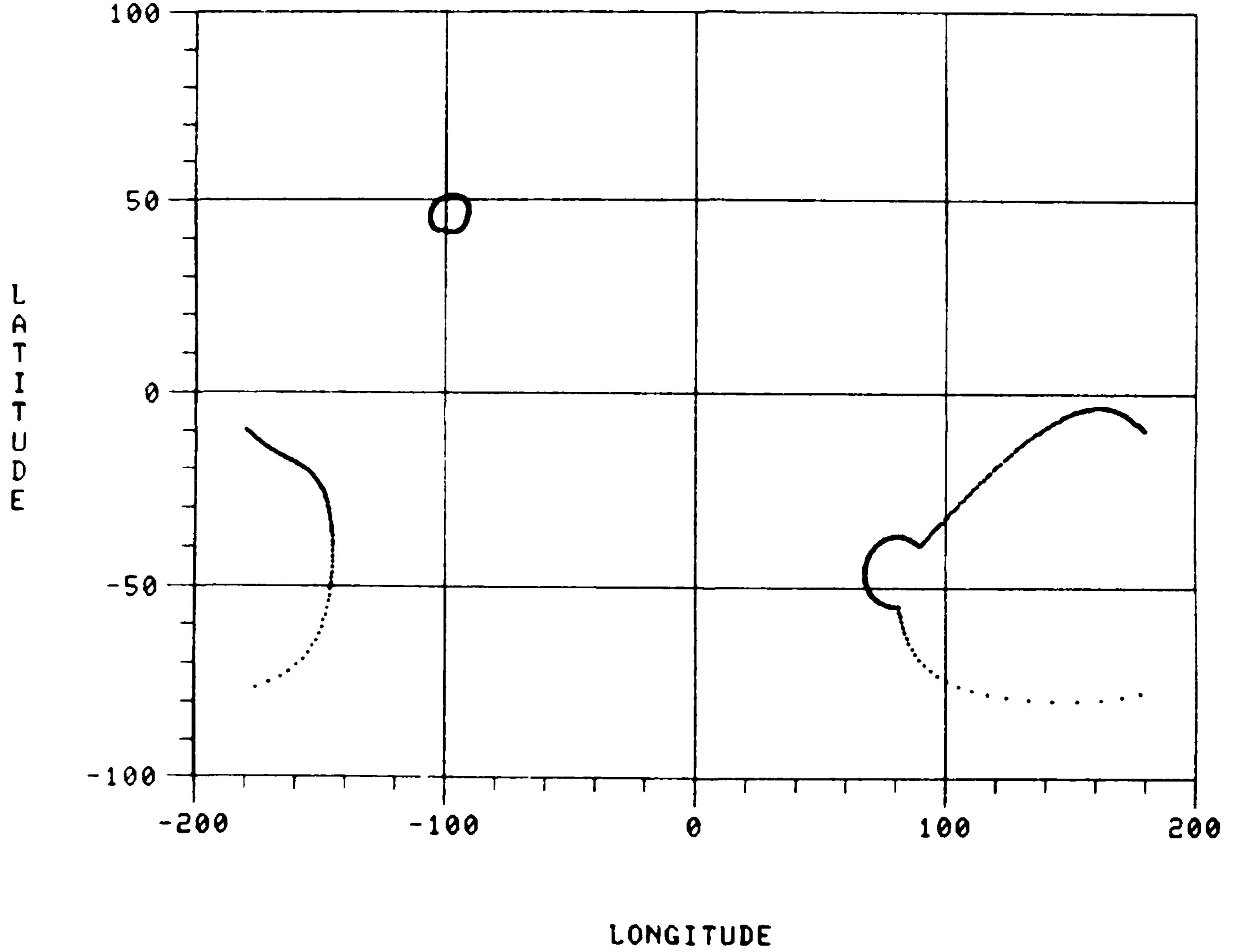


Figure 34.

6-JUL-87

LA REUNION NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL

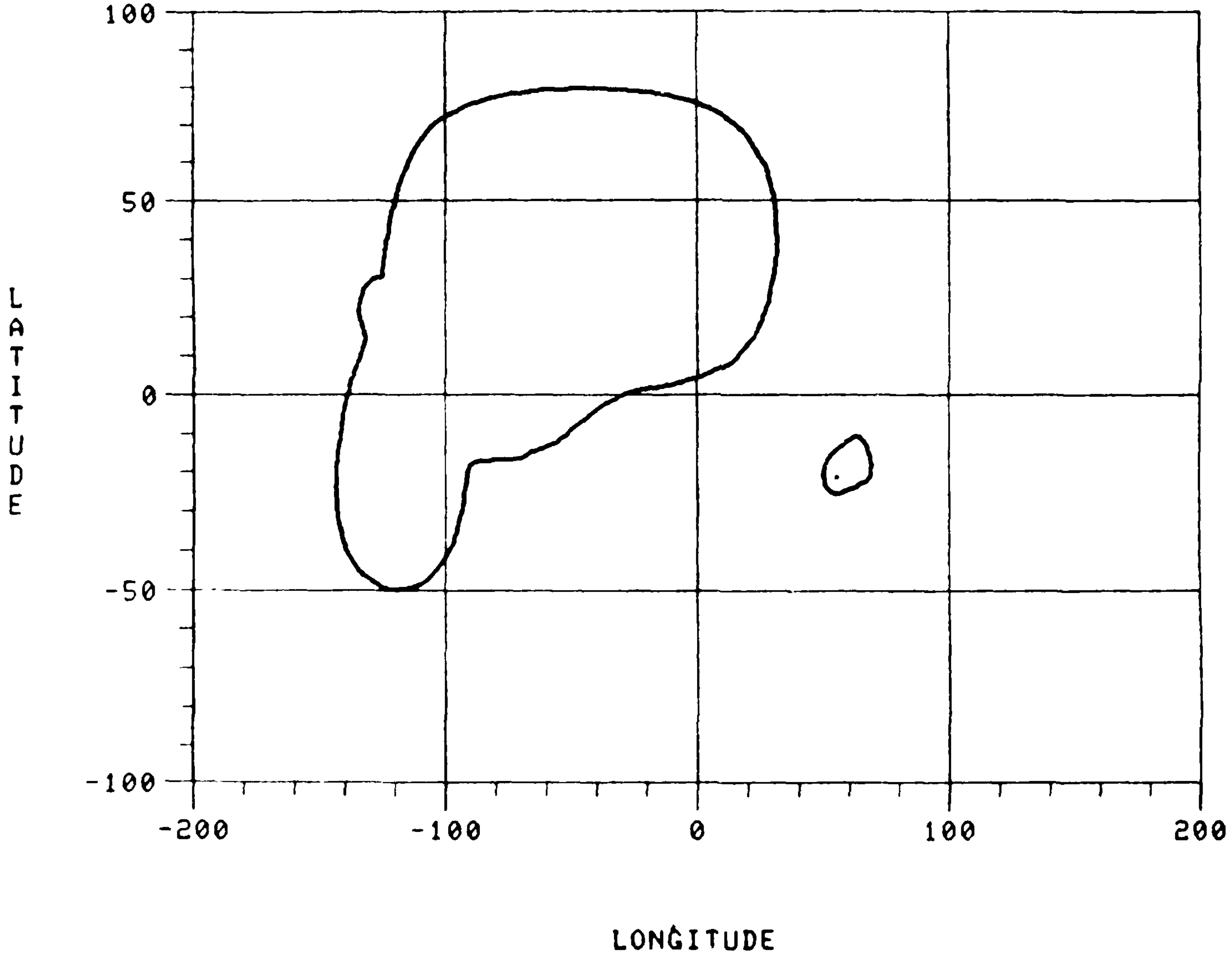


Figure 35.

6-JUL-87

LA_REUNION NIGHTTIME MODAL INTERFERENCE: DREO MODEL

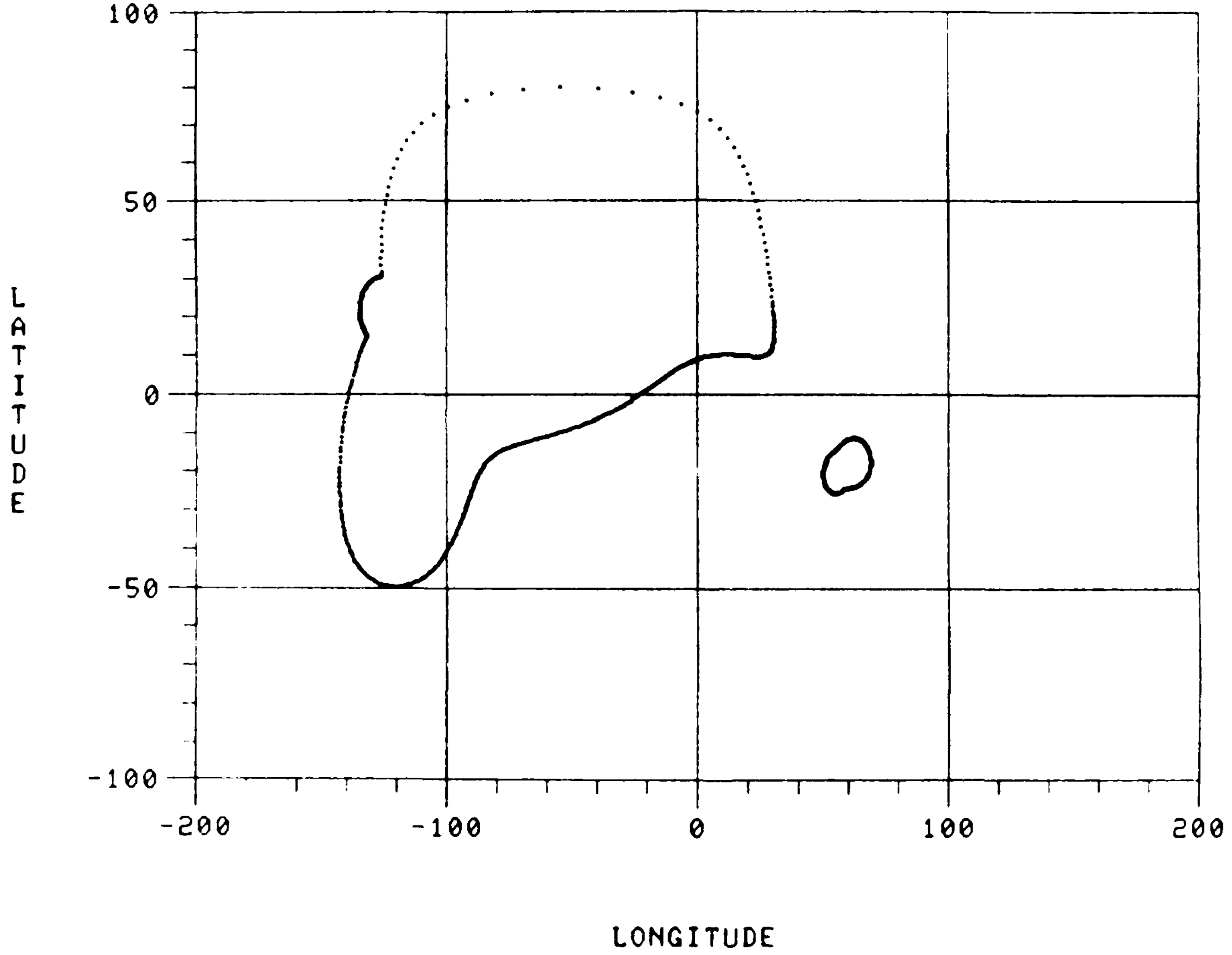


Figure 36.

21-SEP-87

ARGENTINA NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL

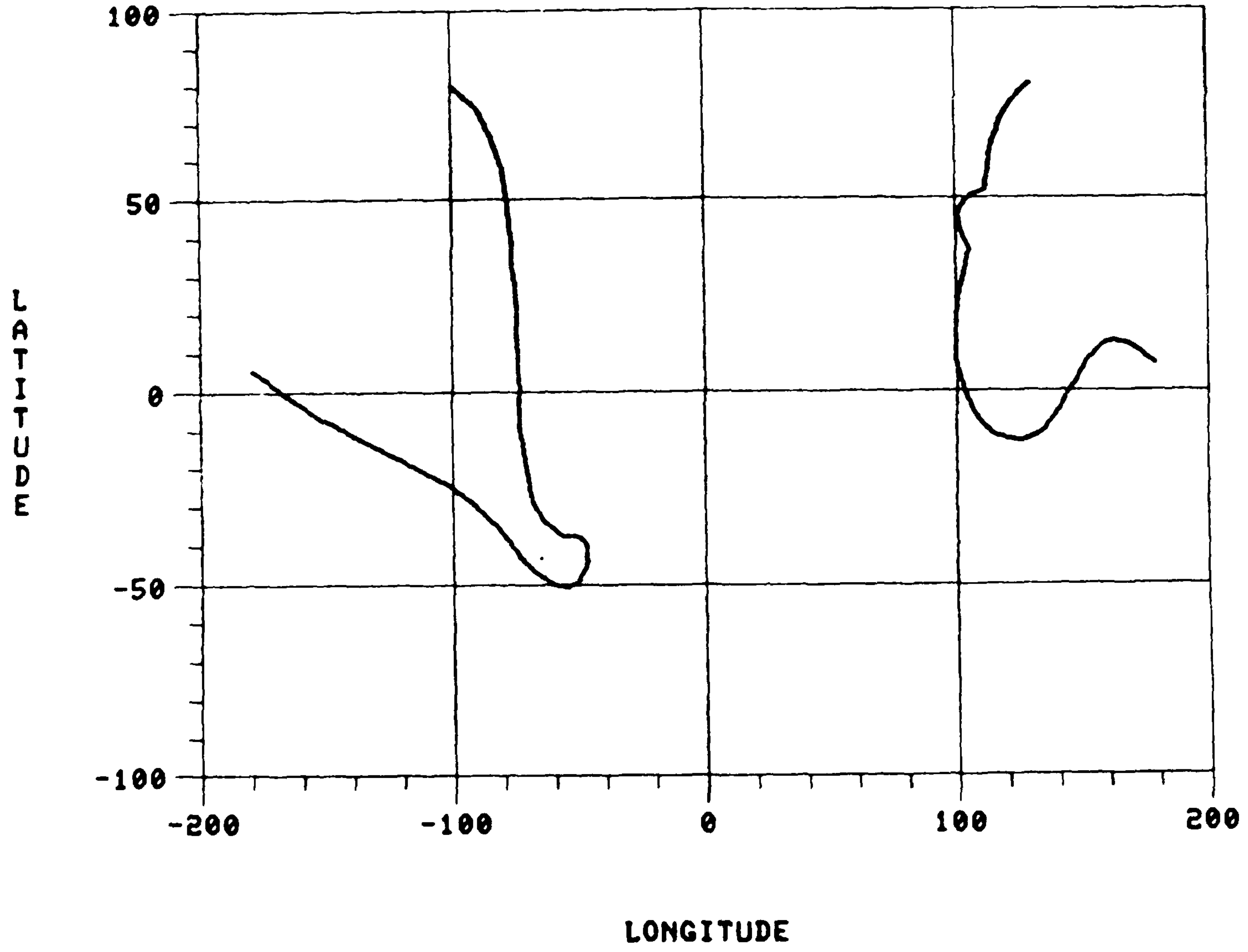


Figure 37.

21-SEP-87

ARGENTINA NIGHTTIME MODAL INTERFERENCE: DREO MODEL

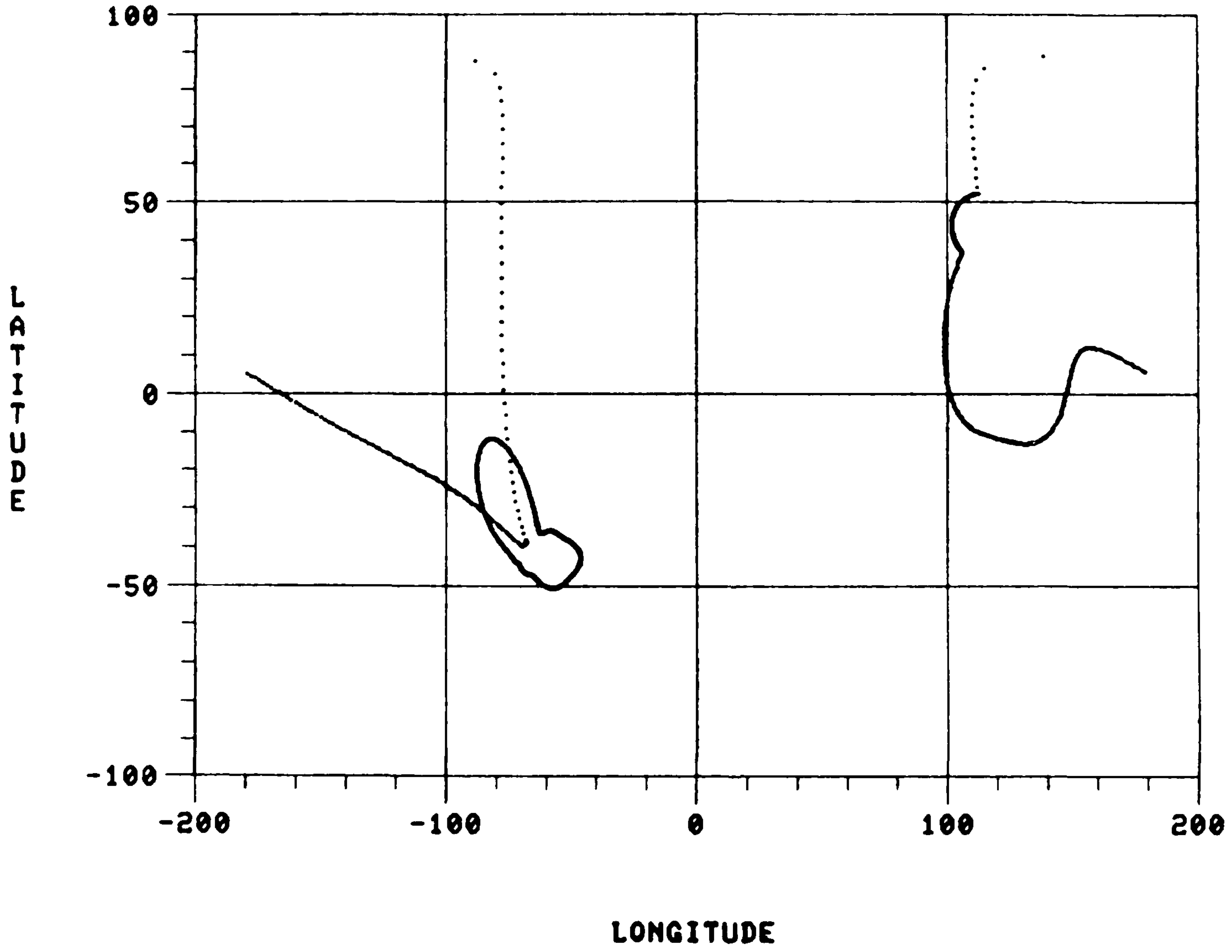


Figure 38.

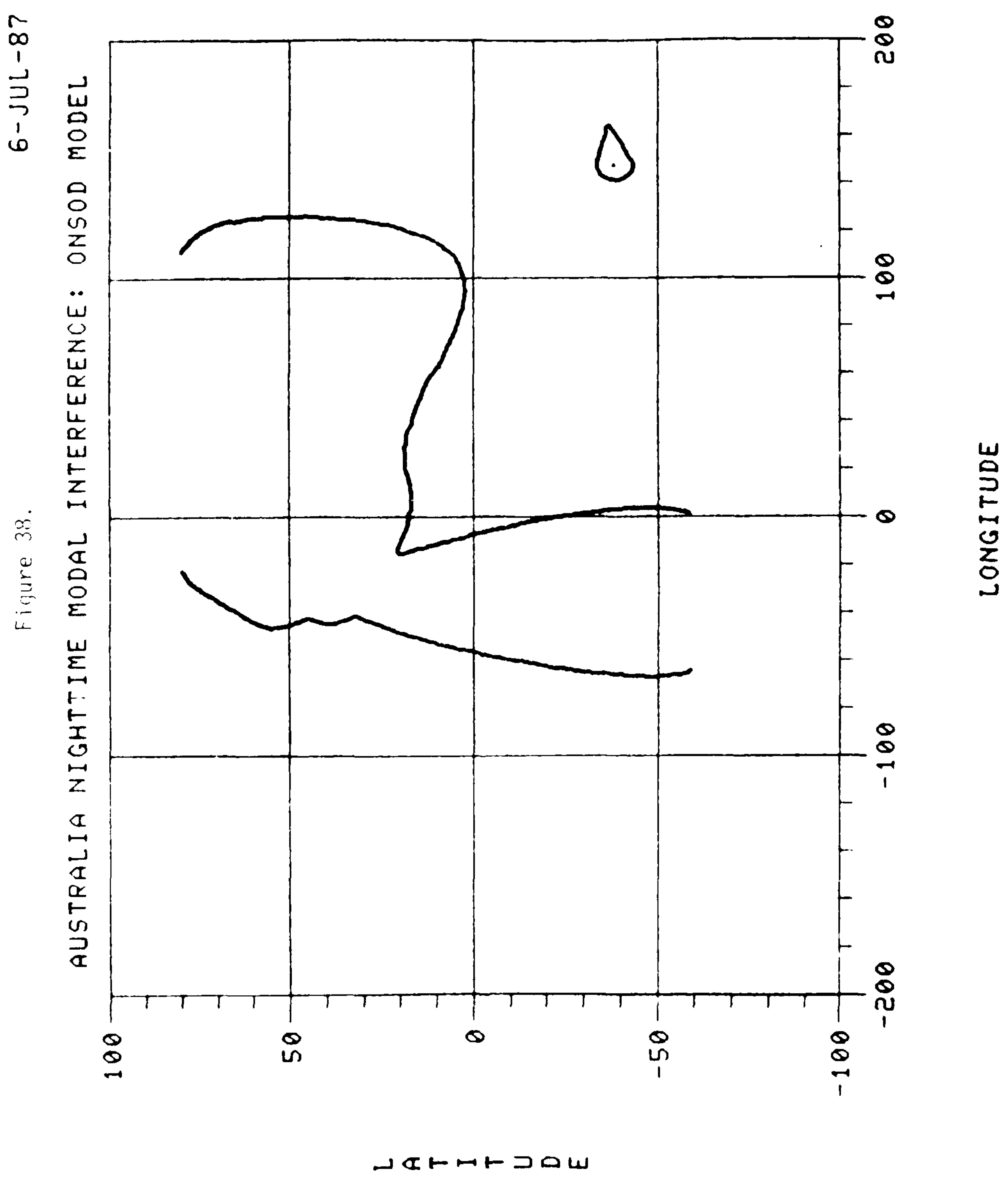

AUSTRALIA NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL
6-JUL-87
LATITUDE
100
50
0
-50
-100
LONGITUDE
-200
-100
0
100
200

Figure 39.

6-JUL-87

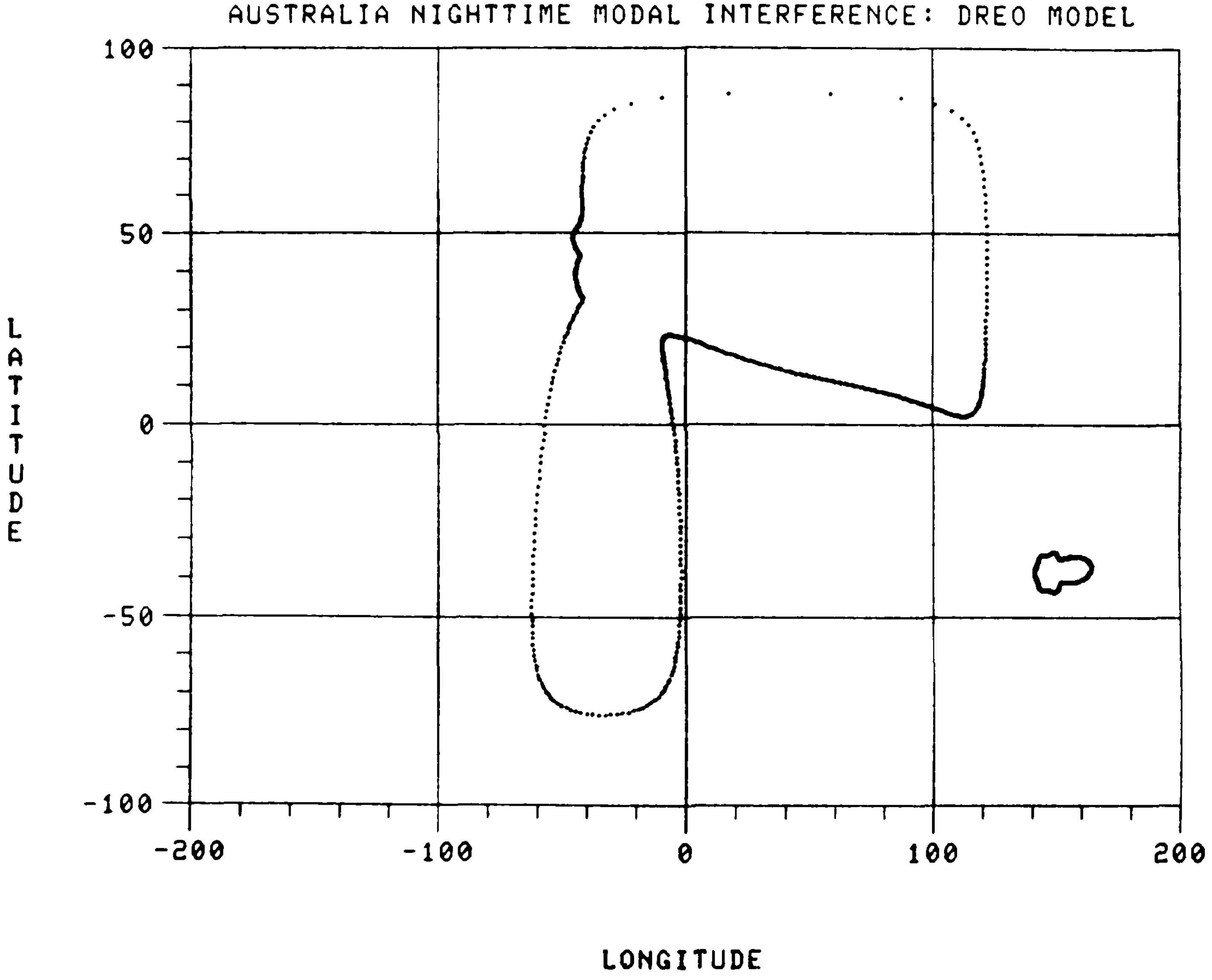


Figure 40.

6-JUL-87

JAPAN NIGHTTIME MODAL INTERFERENCE: ONSOD MODEL

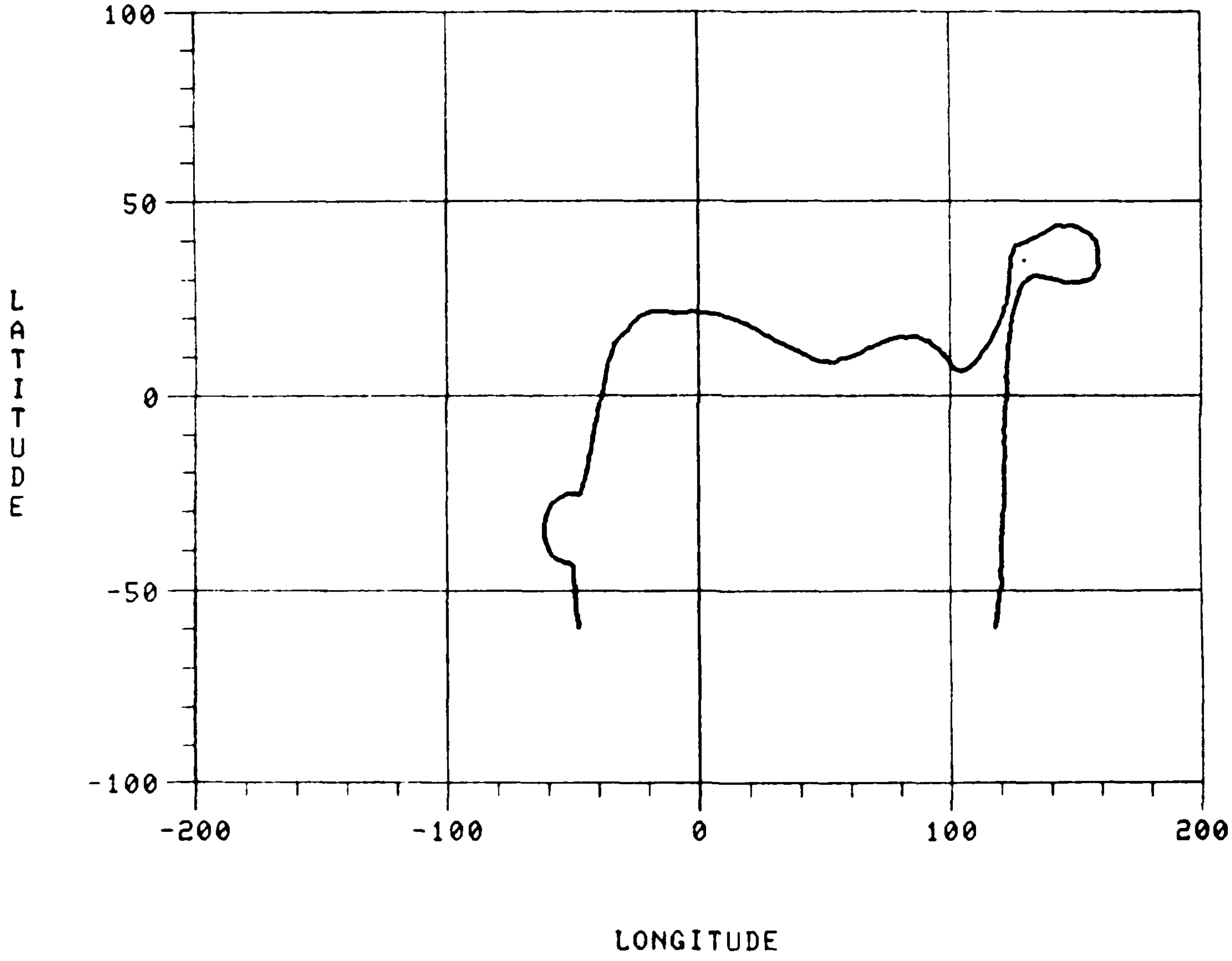
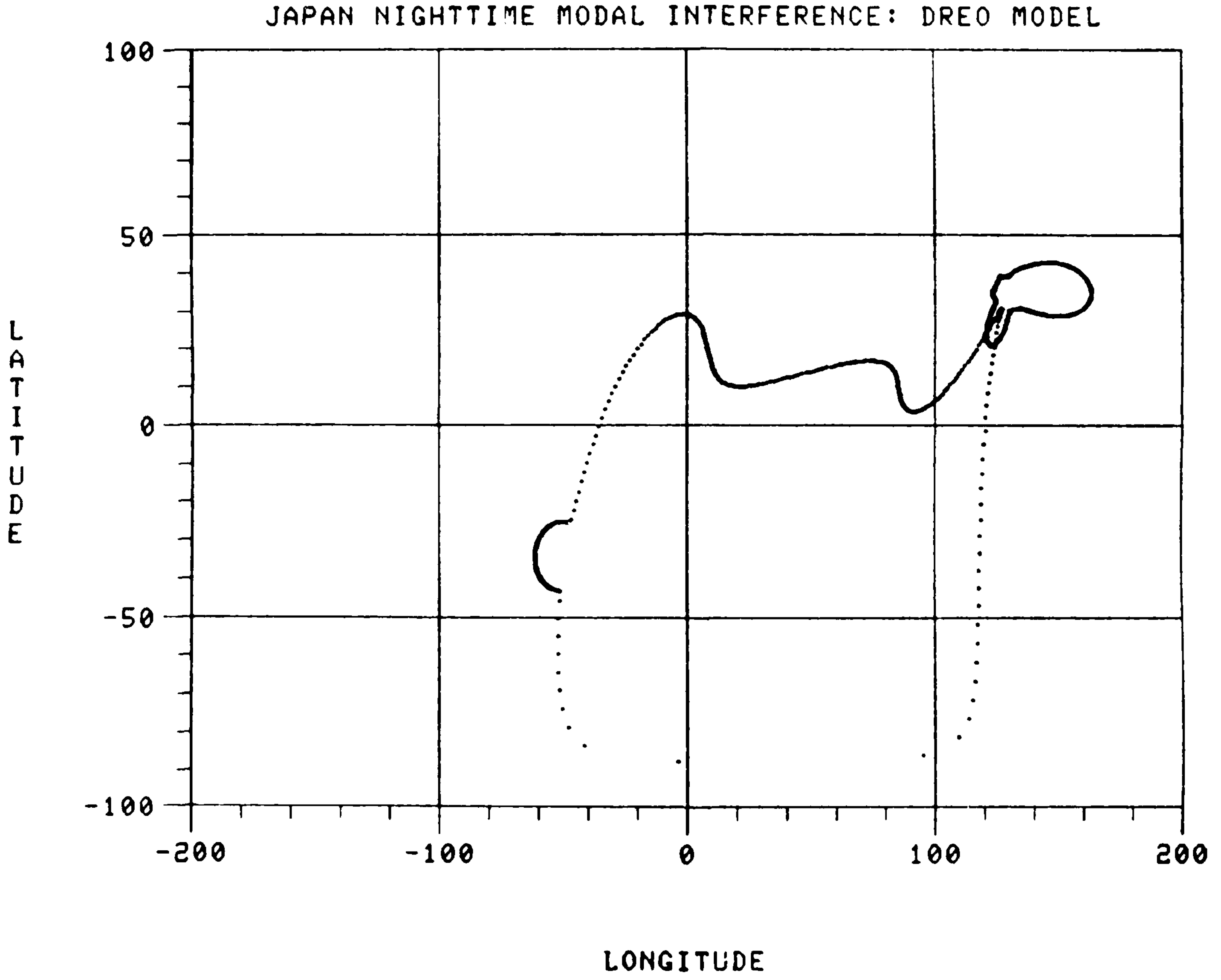


Figure 41.

6-JUL-87



3.2.2 SOLAR EFFECTS

The sun effects Omega modal interference in several significant ways. Firstly the effect described by the nighttime model in the last section is only present if a significant portion of the signal path is in darkness. Secondly, if the signal path crosses the day/night terminator at a high incidence angle (close to normal), then there is a phenomenon known as mode conversion, whereby the primary mode energy can be partially converted into the undesired secondary modes. Thirdly, if the path-terminator angle is very small then there can be a signal reflection off the terminator (which is in effect a physical discontinuity in the ionosphere boundary layer). It has been experimentally determined (reference [12]) that this effect is significant if the path-terminator angle is less than 5°. The first effect is relatively long lasting and will be used in the station selection process. The second effect is not normally significant, but can amplify the first effect. The third effect is relatively short lived and fairly easily determined, and is therefore used as a real time data rejection criterion, rather than for initial selection.

To determine the extent to which the path from the station to the receiver is illuminated by the sun, we simply determine whether either the points 1/3 of the way from each end of the path are illuminated. This is explained as follows:

Assuming that the earth is spherical, then the terminator is a great circle and the (shortest) signal path is a section of a great circle which is less than half. Therefore the path can cross the terminator at most once. Hence if any given point on the path is illuminated then the remainder of the path to one side or the other of this point must also be illuminated.

Now whether a point on the earth's surface is illuminated or not is equivalent to whether or not the vector from the earth's centre to the point in question has a positive or negative projection onto the vector from the earth's centre to the solar subpoint.

SOLAR SUBPOINT:

Now to determine whether or not a point on the earth is illuminated we first find the approximate latitude λ_s and longitude L_s of the solar subpoint, which is a function of the time and date. A simple circular orbit approximation can provide a good initial estimate, assuming the earth's inclination (from the rotation axis to the orbital plane) is $\varepsilon = 0.40913$ radians and the sidereal year is $y = 365.256$ days. This yields (see reference [11] for some derivations):

$$\lambda_s \approx \frac{\pi}{180} \varepsilon \sin\left(2\pi \frac{D-81}{y}\right) \qquad (3a)$$

$$L_s \approx 2\pi \left(\frac{12 - G}{24}\right) - \tan^2\left(\frac{\varepsilon}{2}\right) \sin\left(4\pi \frac{D-81}{y}\right) \qquad (4a)$$

where

D = Julian day
G = GMT in decimal hours

Note that a circular orbit would put the Vernal equinox at about D = 81 (midway between the two solstices).

Comparing this approximation to a much more accurate (and much more complicated) algorithm from the MINS celestial navigation software, indicates that equations (3a) and (4a) are accurate to a few degrees. A simple correction for the orbital eccentricity (see reference [11]) can reduce these erroıs to below one half degree. The improved formulae are:

$$\lambda_s \approx \frac{\pi}{180} \varepsilon \sin\left(2\pi \frac{D-81}{y} + E_1\right) \qquad (3b)$$

$$L_s \approx 2\pi \left(\frac{12 - G}{24}\right) - \tan^2\left(\frac{\varepsilon}{2}\right) \sin\left(4\pi \frac{D-81}{y}\right) + E_1 \qquad (4b)$$

where

$$E_1 = 2e \sin\left(2\pi \frac{D-1}{y}\right) \qquad (5)$$

e = .0167 is the earth's orbital eccentricity. (6)

and the orbital perihelion occurs about January 2 (hence the D-1 in the expression for E_1).

Equations (3b) and (4b) produce an approximation that is correct to within about 0.5 degree, or about 55 km. on the earth's surface. This is quite adequate for station selection purposes, since the regions of modal interference described in section 3.2.1 are not so precisely defined. In fact for the purpose of predicting the location of the day-night terminator this is more than adequate since the non-spherical shape of the earth distorts the terminator anyway, and the terminator is constantly moving, at about 1,667 km/hr at low latitudes. In other words the .5 degree uncertainty represents about 2. minutes in time.

For a given latitude λ and longitude L, the earth centred vector is

$$\mathbf{V} = \begin{bmatrix} \sin\lambda \\ \cos\lambda \sin L \\ \cos\lambda \cos L \end{bmatrix} \tag{7}$$

Now we define this vector for the solar subpoint $\mathbf{V}_s$, the transmitter position $\mathbf{V}_t$ and the receiver position $\mathbf{V}_r$. These three vectors can then be easily used to determine:

1/ whether or not the day/night terminator is on the signal path from the transmitter to the receiver,
2/ whether the signal path is more or less than 1/3 in darkness.
3/ the angle between the terminator and the signal path

First, the day/night terminator crosses the signal path if and only if one path endpoint is illuminated and the other is in darkness. As explained above this is true if and only if:

$$\begin{array}{llll} & \mathbf{V}_s \cdot \mathbf{V}_t > 0 & \text{and} & \mathbf{V}_s \cdot \mathbf{V}_r < 0 \\ \text{or} & & & \\ & \mathbf{V}_s \cdot \mathbf{V}_t < 0 & \text{and} & \mathbf{V}_s \cdot \mathbf{V}_r > 0 \end{array} \tag{8}$$

ILLUMINATION OF SIGNAL PATH:

Secondly, the signal path is more than 1/3 in darkness if and only if either of the points on the path 1/3 of the distance from each end is in darkness, which is true if and only if

$$\mathbf{V}_s \cdot (\mathbf{V}_t + 2\mathbf{V}_r) < 0 \qquad \text{or} \qquad \mathbf{V}_s \cdot (\mathbf{V}_r + 2\mathbf{V}_t) < 0 \tag{9}$$

TERMINATOR - SIGNAL PATH ANGLE:

Thirdly, the angle β between the terminator and the signal path is the angle between the two planes through the earth's centre defined by:

-the solar subpoint (ie. the plane normal to $\mathbf{V}_s$)
-the station and transmitter (ie the plane normal to $\mathbf{V}_t x \mathbf{V}_r$)

The angle between these two planes is the angle between their normal vectors, so that

$$\cos\beta = \mathbf{V}_s \cdot (\mathbf{V}_t x \mathbf{V}_r) / |\mathbf{V}_t x \mathbf{V}_r| \qquad (10)$$

Once it has been determined that the terminator does cross the signal path, this simple vector product can then be compared to the appropriate threshold to determine whether or not β is too small i.e.

$$\cos\beta > .9962 \simeq \cos(5°) \qquad (11)$$

or

$$|\beta| < 5° \qquad (12)$$

When testing for very small incidence angles the better approximation (3b) and (4b) should be used to find $\mathbf{V}_s$.

A simple geometric consideration will show that this cannot be true unless

$$|\mathbf{V}_s \cdot \mathbf{V}_r| \leq \sin(5°) \qquad (13)$$

which implies that the receiver is within 5° (at the earth's centre) of the terminator. *This fact* provides *a single simple* preliminary test, (13), that allows full testing (using (10) on all stations) to be done only near dawn and dusk.

4.0 EXPECTED PHASE ERROR

The expected phase errors can be approximated by using signal to noise information. Since they are being used for comparison only, the relative size of the errors is sufficient (a constant scale factor on all phase errors will not effect the selection decision). The Omega receiver that we are currently using is an MX1105, which provides S/N ratios for all 8 stations as a number from 0.00 to 0.99. Since it is the signal phase rather than the amplitude that is used by the receiver (in a phase locked loop), and since most of the phase error is due to signal propagation irregularities (described stochastically in reference [1] and physically in reference [9]) rather than receiver phase tracking noise, the relationship between S/N ratio and phase error is not particularly strong. The nominal phase error is expected to be equivalent to roughly 2000 metres, only a few hundred metres of which is noise. Theoretically the standard deviation of the phase error due to signal noise (not due to propagation errors) should be proportional to $1/\sqrt{(S/N)}$ as shown in reference [12]. Since the the MX1105 does not provide the actual S/N, (or a simple function of S/N) we will model the expected phase error, from all sources, using a very simple ad-hoc expression:

$$\Delta\omega \simeq 1600 + 400/\rho \tag{14}$$

where ρ is the S/N ratio parameter from the receiver. As was described in section 3.1, no Omega station will be used for which $\rho < .10$, so the above expression (14) for $\Delta\omega$ ranges from 2004 to 5600 metres and has no singularity. The contribution due to receiver noise is therefore approximated by

$$f(\rho) \equiv 400/\rho \tag{15}$$

Table II below indicates how this expression compares to the theoretical values. Note that theoretically the phase error should be proportional to $1/\sqrt{(S/N)}$ where relationship between ρ and signal to noise dB was listed in section 3.1. and of course

$$dB \equiv 10\log_{10}(S/N) \tag{16}$$

From this table we see that for $\rho > .90$ the approximation $f(\rho)$ is not very close, but in this range it has negligible contribution to the estimate of $\Delta\omega$ anyway, and can be safely ignored. Another consideration is that phase noise is not entirely due to signal noise. Atmospheric effects also introduce phase noise that must be included in our model, and our experience has shown that in fact $f(\rho)$ is more realistic than $500/\sqrt{S/N}$. Therefore for the purpose of predicting the expected magnitude of phase error equations (14) and (15) should be quite realistic over the full range of S/N ratio.

TABLE II. Expected Phase Error Due to Signal Noise

ρ	$f(\rho)$	dB $10\log_{10}(S/N)$	S/N	Theoretical Value $500/\sqrt{S/N}$
0.03	13333.	-30.	.001	16000.
0.10	4000.	-19.	.013	4400.
0.20	2000.	-13.	.050	2250.
0.30	1333.	- 8.	.16	1250.
0.40	1000.	- 6.	.25	1000.
0.50	800.	- 4.	.40	800.
0.60	667.	- 2.	.63	650.
0.70	571.	- 1.	.79	550.
0.80	500.	3.	2.0	355.
0.90	444.	10.	10.	160.
0.95	421.	16.	40.	80.
0.97	412.	20.	100.	50.
0.98	408.	30.	1000.	16.
0.99	404.	40.	10000.	5.

5.0 ENUMERATION

There are two aspects of enumeration. First, once the available stations have been identified, as described in chapter 3, then sets of $n \leq 5$ stations (less if 5 are not available) must be selected. Then, given these n stations, the possible sets of n-1 (≤ 4) linearly independent LOPs must be enumerated.

If 5 or fewer stations are locked onto, have not been deselected and are not subject to modal interference, then there is no need to select stations at all. If the maximum of 7 stations are available then the number of possible choices of stations will be

$$7|5 = \frac{7!}{5!2!} = 21$$

Similarly if 6 stations are available then there are only 6 choices of sets of 5.

The more difficult task is to enumerate all linearly independent LOPs, given n stations ($n \leq 5$). Fortunately this can be done offline. The total number of different LOP selections (ignoring linear independence at first) from n stations can be found as follows: the number of different LOPs is $n|2$ and then the number of different sets of (n-1) of these LOPs is

$$\boxed{(n|2)|(n-1)} \tag{17}$$

For 5 stations this is

$$\begin{aligned}(5|2)|4 &= \left(\frac{5!}{3!2!}\right)\Bigg|4 \\ &= 10|4 \\ &= \frac{10!}{6!4!} \\ &= 210\end{aligned}$$

Similarly if there are only 4 stations available then there are only

$$(4|2)|3 = 6|3 = 20$$

choices. If there are only 3 stations, then there are a total of

$$(3|2)|2 = 3$$

choices, and if there are 2 stations then there is only one choice.

Thus the number of different sets of LOPs that must be considered when m stations are available is the number of different subsets of 5 stations (or m stations if m < 5), times the number of different LOPs that can be formed from each subset. These can be listed as follows:

TABLE III. Number of LOP Sets to be Enumerated

available stations	subsets of size n where n=min(5,m)		LOP choices per subset		total number of candidate sets of LOPs	number of linearly independent sets	
m	m|n		(n|2)|(n-1)				
7	21	x	210	=	4410	2625	(21x125)
6	6	x	210	=	1260	750	(6x125)
5	1	x	210	=	210	125	
4	1	x	20	=	20	16	
3	1	x	3	=	3	3	

Now the linear independence of these LOP choices can be analysed without knowing what the stations are, and so can be examined generically ahead of time to reduce the 210, 20 or 6 choices (when n = 5, 4 or 3 respectively). This can be done by testing the rank of the **D** matrix relating stations to LOPs defined by equation (20) of section 6.1 below. This in turn can be done by using Gaussian elimination (with pivoting) to row reduce D, and then test the diagonal elements of the row reduced matrix. It will be full rank if and only if all diagonal elements are non-zero. Since D is composed of plus or minus ones and zeros, there should be no numerical difficulty in determining rank. This has been done, and the results are given in table III.

In the worst case the 210 LOP choices was only reduced to 125, and we do not want to store this many selections, therefore an efficient test is desired for real time implementation. A simple test that n LOPs are linearly independent is that they make use of at least n+1 different stations. This is a necessary but not a sufficient condition for linear independence. Fortunately this simple test catches all but 10 of the linearly dependent selections for the $m \geq 5$ case, reducing the 210 LOPs to 135. Since it is more efficient to evaluate the accuracy of these 10 "extra" sets of LOPs than to use Gaussian elimination on all 135 sets, we can dispense with the Gaussian elimination in the real time implementation, except for the final selection.

The linearly independent sets then must be evaluated based upon their expected positional accuracy, as described in the next section.

6.0 EXPECTED POSITION ERROR

The criterion used, to choose from among the station selection possibilities enumerated, is naturally the resulting expected position accuracy (item 4 of section 2.1). The accuracy of the phase measurements from individual stations (item 2a) is fairly straightforward, and has been dealt with in section 4.0 above. How these phase errors relate to the resulting position error depends on how the position is formed. Here we will consider 2 methods: least squares and Kalman filtered.

In section 6.1 below we will examine in detail how one can express the position accuracy as a function of the phase accuracies, assuming that the position solution is found using a least-squares approach. The accuracy of an unweighted least squares solution depends only on the geometry, so that the relationship between phase error and position error is often called the geometric dilution of precision (GDOP). The example given in section 7.3 illustrates how the least squares position accuracy varies with different station selections and LOP choices, at a particular place and time.

In section 6.2 we examine the effect of a multi-LOP update on a Kalman filter position estimate, as would be performed in MINS. This, as we will see, can be very different from the least squares results. The Kalman filter analysis in fact proves to be simpler and much more efficient to implement than the least squares analysis, but is less illustrative. Section 8.1 illustrates how the station selections used in the least squares example of section 7.3 have different effects on the Kalman filter position error covariance through the filter update.

6.1 GEOMETRIC DILUTION OF PRECISION

To specify this error we first define the position error vector to be

$$\mathbf{X} \equiv \begin{bmatrix} dN \\ dE \end{bmatrix} = \begin{bmatrix} \text{North error} \\ \text{East error} \end{bmatrix} \quad \text{in metres} \qquad (18)$$

We will assume that this position error arises from taking a least squares solution from n = 4 or less Omega LOPs, using the phase information from (n+1) stations. There are a total of 8 Omega stations to choose from, so we define their locations:

$$\begin{aligned} \Omega L &\equiv \text{vector of 8 Omega station latitudes} \\ \Omega\lambda &\equiv \text{vector of 8 Omega station longitudes} \end{aligned} \tag{19}$$

We also define the phase error of the 8 signals (at the receivers location) to be:

$$\xi \equiv \text{vector of 8 Omega phase errors.} \tag{20}$$

Assume a given selection of n Omega LOPs using (n+1) stations. The station selection is defined by a vector S of pointers into ΩL and $\Omega\lambda$ (since we are programming in the "C" language we will use zero indexing):

$$S_i \equiv i^{th} \text{ selected station} \quad (i = 0 \text{ to } n) \tag{21}$$

(so S is a vector of length (n+1), whose components have values of 0 to 7, and are used to point into ΩL, $\Omega\lambda$ and ξ). We then define the vector of the phase errors of the n+1 selected stations:

$$\Phi \equiv \text{vector of n+1 selected Omega phase errors.} \tag{22}$$

so that

$$\Phi_i = \xi_{S_i} \qquad \text{for } i = 0 \text{ to } n \tag{23}$$

The n LOPs are then selected from pairs of these stations. This LOP selection can be defined by an nx2 array of pointers P into the station vector S:

for i = 0 to (n-1).

$$\left\{ \begin{aligned} P_{i0} &\equiv j \qquad \text{if LOPi is formed from stations j minus k} \\ &\qquad\quad\;\, \text{of the selected n+1 vector } \Phi \\ P_{i1} &\equiv k \qquad (\text{or } S_j \text{ minus } S_k \text{ in the total 8 vector } \xi) \end{aligned} \right. \tag{24}$$

Thus the components of P (j and k) take on values from 0 to n and point to components of S. In this way S and P can be used to obtain the LOP error vector

$\mathbf{Z}$ from the phase errors $\boldsymbol{\xi}$, or more simply $\mathbf{P}$ can be used to obtain them directly from $\boldsymbol{\Phi}$. This indexing method is demonstrated by example in section 6.1 below. The vector of LOP errors is defined as:

$$\mathbf{Z} \equiv \begin{bmatrix} \text{LOP1 error} \\ \text{LOP2 error} \\ \text{LOP3 error} \\ \text{LOP4 error} \end{bmatrix} \qquad (25)$$

and so

$$Z_i = \xi_{S_{P_{i0}}} - \xi_{S_{P_{i1}}} \qquad \text{for } i = 0 \text{ to } n-1 \qquad (26)$$

or

$$Z_i = \Phi_{P_{i0}} - \Phi_{P_{i1}} \qquad \text{for } i = 0 \text{ to } n-1 \qquad (27)$$

The latitude or longitude coordinates of the Omega stations used in each LOP can also be obtained in a similar manner, by substituting $\boldsymbol{\Omega L}$ or $\boldsymbol{\Omega\lambda}$ in place of $\boldsymbol{\Phi}$ above.

Equation (27) can be used to define the nx(n+1) transformation matrix $\mathbf{D}$ relating the selected phase error vector $\boldsymbol{\Phi}$ to the LOP error vector $\mathbf{Z}$.

$$\boxed{\mathbf{Z} = \mathbf{D}\,\boldsymbol{\Phi}} \qquad (28)$$

where

$$D_{ij} = \delta_j^{P_{i0}} - \delta_j^{P_{i1}} \qquad (29)$$

where the Kronecker delta function is:

$$\delta^i_j \equiv \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{30}$$

In other words:

$$\mathbf{D}_{ij} = \begin{cases} 1 & \text{if } j = \mathbf{P}_{i0} \\ -1 & \text{if } j = \mathbf{P}_{i1} \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

The example in section 6.1.1 below illustrates a typical **D** matrix, with corresponding **P** matrix and **S** vector.

For LOP errors that are small compared to the inter-transmitter distances, the LOPs are linear and the error in the LOPi (line of position using the phase difference of station $\mathbf{S}_{\mathbf{P}_{i0}}$ minus station $\mathbf{S}_{\mathbf{P}_{i1}}$), in metres, is linearly related to the north and east position error by a fairly simple geometric relationship (see figure 3.6 of reference [1]). To express this we must first define the vector of bearing angles to the Omega stations from the receiver:

$$\boldsymbol{\Psi} \equiv \text{vector of 8 bearing angles to Omega stations} \tag{32}$$

The bearings to the selected stations are therefore:

$$\boldsymbol{\Psi}_{\mathbf{S}_i} = \text{bearing to the } i^{th} \text{ selected station} \tag{33}$$

Then the position error vector **X** at the receiver is related to the LOP error vector **Z** by the nx2 transformation matrix **H** as follows:

$$\boxed{\mathbf{Z} \;=\; \mathbf{H}\,\mathbf{X}} \tag{34}$$

where for i=0 to (n-1)

$$\mathbf{Hi0} \;\equiv\; \cos\mathbf{\Psi}_{\mathbf{S}_{\mathbf{Pi1}}} - \cos\mathbf{\Psi}_{\mathbf{S}_{\mathbf{Pi0}}} \tag{35}$$

$$\mathbf{Hi1} \;\equiv\; \sin\mathbf{\Psi}_{\mathbf{S}_{\mathbf{Pi1}}} - \sin\Psi_{\mathbf{S}_{\mathbf{Pi0}}} \tag{36}$$

so that

$$\mathbf{Z}_i \;=\; \left[\;(\cos\mathbf{\Psi}_{\mathbf{S}_{\mathbf{P}_{i1}}} - \cos\mathbf{\Psi}_{\mathbf{S}_{\mathbf{P}_{i0}}}) \;\; (\sin\mathbf{\Psi}_{\mathbf{S}_{\mathbf{P}_{i1}}} - \sin\Psi_{\mathbf{S}_{\mathbf{P}_{i0}}})\;\right] \begin{bmatrix} dN \\ dE \end{bmatrix} \tag{37}$$

Now from (28) and (34) we can solve for the expected least squares position error **X** as a function of the expected phase errors **Φ**, as follows. Eliminating **Z** from (28) and (34) yields:

$$\mathbf{HX} \;=\; \mathbf{D\Phi}$$

$$\therefore \qquad (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\,\mathbf{HX} \;=\; (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\,\mathbf{D\Phi}$$

$$\therefore \qquad \mathbf{X} \;=\; (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\,\mathbf{D\Phi} \tag{38}$$

(which is the unweighted least squares solution). This can be written as:

$$\boxed{\mathbf{X} = \mathbf{F}\,\boldsymbol{\Phi}} \tag{39}$$

where

$$\mathbf{F} \equiv (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{D} \tag{40}$$

Note that the $(\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T$ portion of $\mathbf{F}$ is the generalised inverse of $\mathbf{H}$, which leads to the least squares solution.

Now the expected radial position error is $\sqrt{J}$, where

$$\begin{aligned} J &\equiv E\{\mathbf{X}^T\mathbf{X}\} \qquad\qquad (= dN^2 + dE^2) \qquad (41) \\ &= \mathrm{trace}(\ E\{\mathbf{X}\mathbf{X}^T\}\ ) \\ &= \mathrm{trace}(\ E\{\mathbf{F}\boldsymbol{\Phi}\boldsymbol{\Phi}^T\mathbf{F}^T\}\ ) \\ &= \mathrm{trace}(\ \mathbf{F}\ E\{\boldsymbol{\Phi}\boldsymbol{\Phi}^T\}\ \mathbf{F}^T\ ) \end{aligned}$$

so that

$$\boxed{J = \mathrm{trace}(\ \mathbf{F}\mathbf{C}\mathbf{F}^T\ )} \tag{42}$$

where

$$\mathbf{C} \equiv E\{\boldsymbol{\Phi}\boldsymbol{\Phi}^T\} \tag{43}$$

is the expected covariance of the phase errors. Note that $\mathbf{F}\mathbf{C}\mathbf{F}^T$ in (42) is a 2x2 matrix, so that the trace is just the sum of the 0,0 and 1,1 elements.

The number of computations needed to evaluate J (once $\mathbf{H}$ and $\mathbf{D}$ have been evaluated) can be easily found as follows:

$\mathbf{H}$ is an nx2 matrix

$\mathbf{D}$ is nx(n+1)

$\therefore$ $\mathbf{H}^T\mathbf{H}$ is 2x2

$\mathbf{H}^T\mathbf{D}$ and $\mathbf{F}$ are 2x(n+1)

Therefore to form $\mathbf{H}^T\mathbf{H}$ requires $4n$ multiplies

and $(\mathbf{H}^T\mathbf{H})^{-1}$ requires 6 multiplies

and $\mathbf{H}^T\mathbf{D}$ requires $2n(n+1)$ multiplies

Multiplying these to get **F** requires $4(n+1)$ multiplies

Finally to form J requires another $4(n+1)$ multiplies

For a total of $\mathbf{(2n+12)(n+1) + 2}$ multiplies.

This total of $\mathbf{(2n^2 + 14n + 14)}$ multiplications assumes that **C** is the identity and can be ignored. For n=4 LOPs, this total is 102. This of course must be done for each candidate set of LOPs, as are enumerated in Table 1 of section 3.0. In the worst case (m=7) this results in a total of 2,625 x 102 = 267,750 multiplications.

Determining the value of **C** is the subject of item 2a of section 2.1, and was dealt with in chapter 4. It should be noted however that **C** is simply a diagonal (n+1)x(n+1) matrix.

6.1.1 EXAMPLE

To illustrate the use of the pointing vectors and matrices defined above, we assume that the Omega stations in use are A,C,D and H, and that the LOPs formed from these stations are:

C A
D G
D H
G H

In this case the **S** vector, defined by (21) is

$$\mathbf{S} = \begin{bmatrix} 0 \\ 2 \\ 3 \\ 6 \\ 7 \end{bmatrix}$$

which clearly indicates which stations are being used (ie. 0≡A, 2≡C, 3≡D etc.). The **P** matrix defined by (24) is

$$\mathbf{P} = \begin{bmatrix} 1 & 0 \\ 2 & 3 \\ 2 & 4 \\ 3 & 4 \end{bmatrix}$$

which explicitly indicated which stations are used for each of the 4 LOPs. Finally the **D** matrix defined by (29) is

$$\mathbf{D} = \begin{bmatrix} -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

which is the transformation matrix from station errors to LOP errors, and also can be seen to illustrate graphically, if somewhat abstractly, how the LOPs are formed from the selected stations.

6.2 KALMAN FILTER COVARIANCE

In this section we will examine in detail how the different choices of stations and LOPs affect (or do not affect) the position accuracy of a Kalman filter, as expressed by the covariance of the position error states. The concepts and notation of section 6.1 are still relevant here, however we must separate the phase error vector $\mathbf{\Phi}$ into its correlated and uncorrelated components:

$$\mathbf{\Phi} \equiv \mathbf{M} + \mathbf{V} \tag{44}$$

where **M** is the time correlated portion (modelled in MINS by Markov processes) and **V** is the uncorrelated portion (modelled by Gaussian white noise) so that equation (28) relating phase errors $\mathbf{\Phi}$ to the LOP errors **Z** becomes:

$$\mathbf{Z} = \mathbf{DM} + \mathbf{DV} \tag{45}$$

We must also realize that the filter has error states **X** that will (attempt to) track and remove the correlated error **DM**, and that the error in the filtered position is not **X** but is represented statistically by the covariance matrix **P**. Therefore in place of equation (34) relating the position error **X** to the LOP error **Z** we have the usual Kalman filter measurement equation relating the position error **estimate** **X** to the LOP error **Z** (note the different meaning for **X** in the filter case)

$$\boxed{\mathbf{Z} = \mathbf{HX} + \mathbf{DV}} \tag{46}$$

where the measurement noise covariance is **R** where

$$\begin{aligned} \mathbf{R} &\equiv E\{ (\mathbf{DV})(\mathbf{DV})^T \} \\ &= \mathbf{DCD}^T \end{aligned} \tag{47}$$

where **C** is the covariance of the phase noise vector **V**, and is assumed to be diagonal (since the different phase errors are uncorrelated) with elements given by equation (15) of chapter 4 as functions of the S/N ratio of the appropriate signal. The measurement matrices H and D of equations (46) are as given in section 6.1, by equations (29), (35) and (36).

Now the Kalman filter measurement equation has been specified by (46) and (47) where $\mathbf{H}$ and $\mathbf{R}$ both depend upon the station and LOP selection. The question is: how do different selections affect the position error covariance. This covariance is affected by the measurement update according to the usual equations (see for example reference [13]):

$$P(+) = [\, I - \mathbf{KH}]\, P(-) \tag{48}$$

where

$$\mathbf{K} = P(-)\, \mathbf{H}^T [\, \mathbf{H}P(-)\mathbf{H}^T + \mathbf{R}\,]^{-1} \tag{49}$$

From these equations it is not clear how to select $\mathbf{H}$ and $\mathbf{R}$ to minimize $\mathbf{P}$. However a matrix inversion theorem can be used to show that

$$\boxed{\mathbf{P}^{-1}(+) = \mathbf{P}^{-1}(-) + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}} \tag{50}$$

(this can be confirmed by verifying that $P(+)P^{-1}(+) = I$). In equation (50) the effect of station and LOP selection is isolated and explicit. In order to minimize P (actually the trace: P(0,0) + P(1,1)) we must maximize $\mathbf{P}^{-1}$, by maximizing

$$\boxed{J \equiv \operatorname{trace}(\, \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\,)} \tag{51}$$

which using (47) in this case becomes

$$J = \operatorname{trace}(\, \mathbf{H}^T [\, \mathbf{DCD}^T\,]^{-1} \mathbf{H}\,) \tag{52}$$

Equation (50) can be used to show that the equivalence of minimizing the trace of $\mathbf{P}$ and maximizing the trace of $\mathbf{P}^{-1}$ is exact if we assume that the a priori latitude and longitude error uncertainties are equal and uncorrelated ($\mathbf{P}$ is diagonal with P(0,0) = P(1,1). This is a perfectly reasonable assumption and is in fact the way in which the MINS covariance is initialized. Choosing the best stations based on this criteria will therefore lead to the optimal selection of Omega stations in the absence of other measurements (so that Omega is expected to provide complete positioning information).

It is possible, however, that the a priori covariance $\mathbf{P}$ is substantially anisotropic during station reselection, after MINS has already been operating with other sensors. For example if Loran-C was already providing good latitude information, (**P**(0,0) << **P**(1,1)) then one may wish to configure Omega to concentrate on providing good longitude information (to reduce **P**(1,1)). This is what would happen if (50) were used to exactly minimize the trace of **P**. This is possible, since the inverse of **P** (a 2x2 matrix) can be expressed explicitly in terms of the elements of **P** and (50) can be used to determine the exact function to be maximized in order to minimize the trace of P. This however is not the approach that we will take for MINS, where we want to make the Omega station selection autonomous.

The correctness of "maximizing" $\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}$ can also be seen, however less explicitly, from a more familiar equation, the matrix Riccati equation (see reference [13]):

$$\dot{\mathbf{P}} = \mathbf{FP} + \mathbf{PF}^T + \mathbf{GQG}^T - \mathbf{PH}^T\mathbf{R}^{-1}\mathbf{HP} \qquad (53)$$

where **F**, **G** and **Q** are independent of the station and LOP selection. Here again we see evidence that maximizing J will generally minimize **P** (note that **P** is positive definite), and that if the a priori **P** is diagonal and **P**(0,0)=**P**(1,1) then again equation (53) shows that maximizing J exactly minimizes the trace of the derivative of **P** and consequently of **P**.

When this new definition of J, given by (51), was tested on several sets of LOPs, it was discovered that for a given set of 5 stations the value of J was independent of the LOP selection! This implies that the Kalman filter position accuracy is completely unaffected by the selection of LOPs, once the 5 stations have been chosen. This somewhat surprising result simply reflects the fact that the Kalman filter extracts all information from the 5 stations no matter how they are paired (as long as the LOPs selected are linearly independent). This makes the selection process much more efficient for a Kalman filter since the accuracy calculation has only to be calculated once for each station selection, (rather than 135 times when 5 stations are available).

Chapter 8 describes how this criterion is used by MINS to select the best stations and LOPs, and gives an illustrative example.

7.0 LEAST SQUARES SELECTION

This chapter describes in detail how one could optimally choose Omega stations and LOPs for a least squares position fix, where the errors are as described in section 6.1 above. Since a full enumeration of all possible LOP selections is required in this situation, the selection process is performed in two stages, the first of which narrows the possibilities to 8.

7.1 PRELIMINARY SELECTION

The first stage of the actual selection process is item 5/ of section 2.1, which is to choose from among the choices enumerated (as described in chapter 5), a small number of LOP selections with the best expected accuracies, (determined by the GDOPs weighted by the S/N ratio, as described in chapter 6.1). The number of selections maintained by the preselection stage was chosen to be 8 (or less if 8 are not available). With only one or two stations available there is no selection to make, and with 3 stations available there are only 3 choices of LOPs. However, as seen in table III of chapter 5, as the number of available stations increases beyond 4 the number of LOP selections increases dramatically, in which case the 8 preliminary selections will all have very similar expected accuracies.

Since a simplified test was used to eliminate linearly dependent sets of LOPs, it is possible that some of these preselections are not linearly independent. They are therefore tested fully (using Gaussian elimination as described in chapter 5) before making the final selection. The final decision is then made by applying reliability and efficiency criteria to the remaining selections, as described in section 7.2 below.

Figure 42 of section 7.2 illustrates the entire selection process as a simplified flow chart. The first page of figure 42 is the preliminary selection, and the second page is the final selection.

7.2 FINAL SELECTION

As described in item 6/ of section 2.1, the final selection of LOPs is made by applying two additional tests to the preliminary selection described above. These are intended to provide a selection that is more robust in the face of station loss, and to improve the efficiency when reselecting each hour by avoiding unnecessary reconfiguration.

The first of the two final selection processes is to eliminate, if possible, those selections that use the same Omega station in 3 or more LOPs. This is to avoid excessive dependence upon a single Omega station, because if that signal is lost for whatever reason then all LOPs using that station will be lost. This improves the reliability by guarding against a serious sudden loss of precision. This is implemented by simply increasing by a factor of 1.2 the expected position error J (as determined by the S/N and GDOP through equation (42) of chapter 6) of each of the preliminary selections in which one station appears in 3 or more LOPs, and then reordering them by their new expected errors. Except in extreme situations this 20% boost is generally enough to ensure that such a selection is not used.

The last selection process only applies if MINS is already running with a previously chosen selection of best Omega LOPs and and we wish to determine whether or not a change should be made. Since there is often a negligible difference between the expected accuracies of the few best selections, and since our "expected accuracies" J are only approximations, it may not be worthwhile changing stations, even when the presently used selection is not the very best. This is because a change in stations will require new PPCs (Phase Propagation Corrections) to be calculated before the new station's measurements can be used, and this requires considerable processing time. Rearranging the existing stations to form different LOPs however is quite easily done.

This last selection process begins by testing whether or not the previously selected stations are all still available. If they are not then we simply abandon the previous selection and use the new one. If the currently used station are all still available however, then we estimate the accuracy of this selection. If the new selection gives an improvement of more than 15% then again we use the new selection. If on the other hand the current selection is practically as good (within 15%) as the best selection, then we add it to our preliminary selection list and choose the best selection from this list that does not require a station change. This is illustrated in figure 42 below.

This selection process is to be repeated every hour by MINS. However since the Omega stations are so far apart compared to the distance that a ship can move in one hour, the Omega geometry is unlikely to change very much from hour to hour. Therefore it is quite likely that a station change will not be necessary every hour.

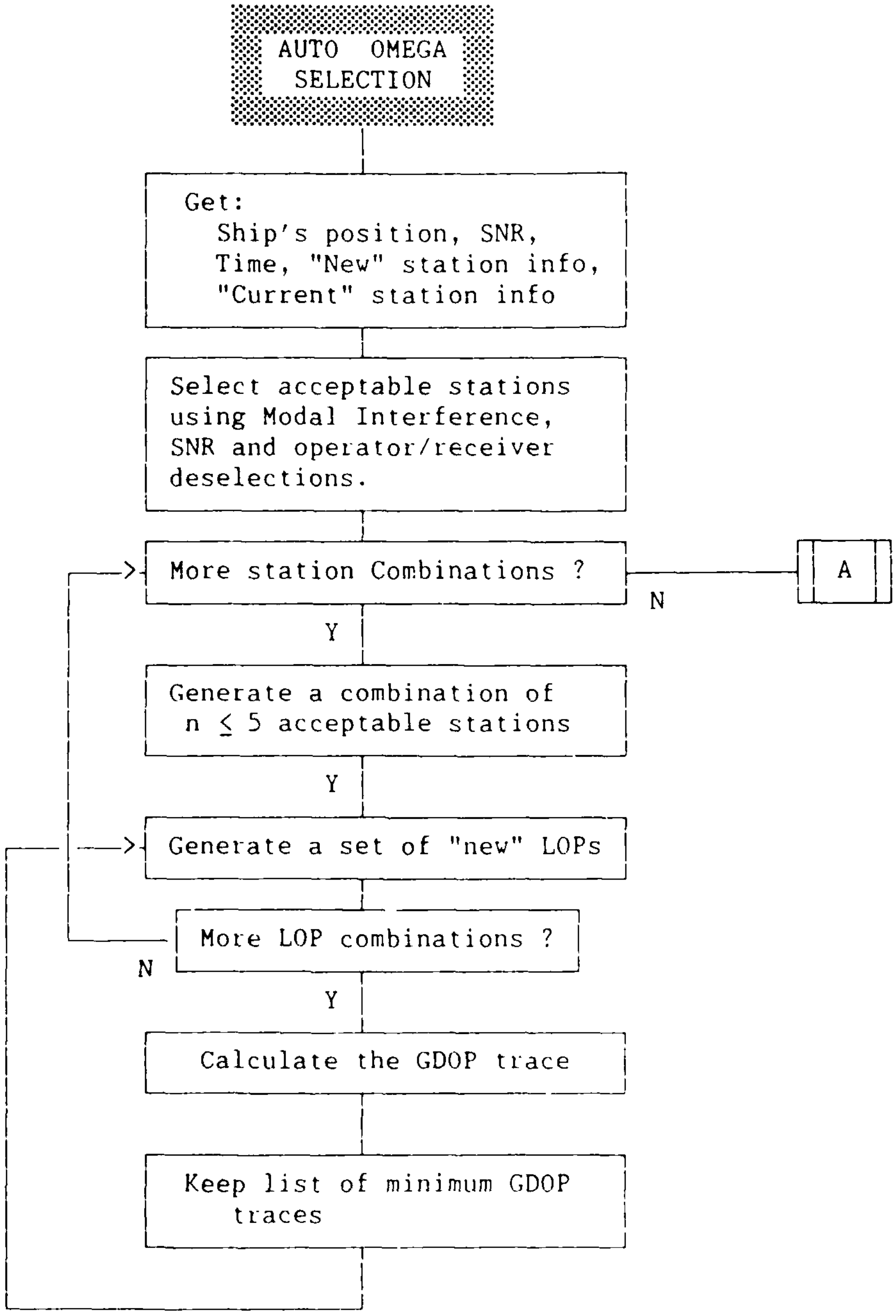

AUTO OMEGA SELECTION
Get: Ship's position, SNR, Time, "New" station info, "Current" station info
Select acceptable stations using Modal Interference, SNR and operator/receiver deselections.
More station Combinations ?
N
A
Y
Generate a combination of n ≤ 5 acceptable stations
Y
Generate a set of "new" LOPs
More LOP combinations ?
N
Y
Calculate the GDOP trace
Keep list of minimum GDOP traces

Figure 42 Flow Diagram for Least Squares OMEGA Station Selection

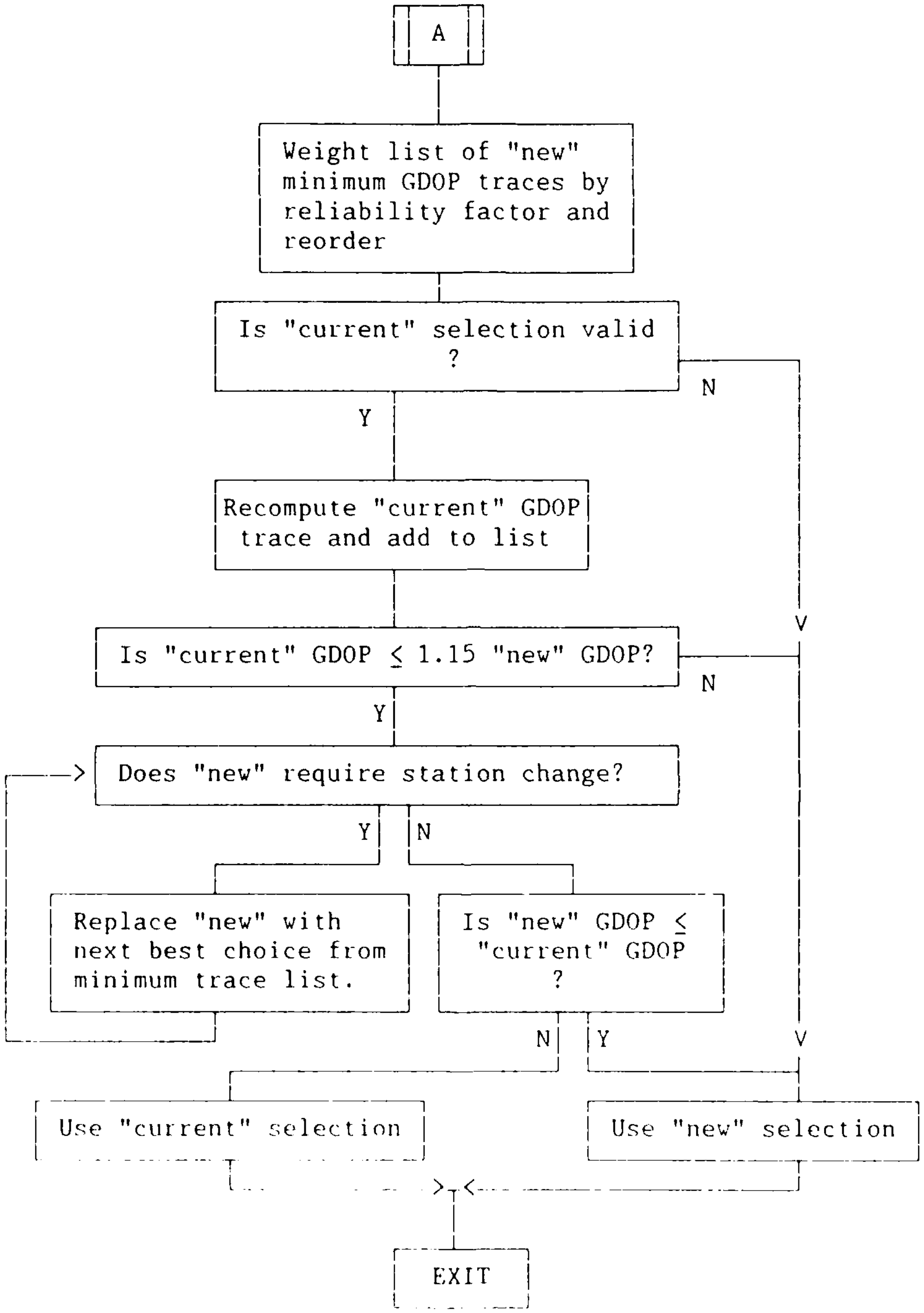

A
Weight list of "new" minimum GDOP traces by reliability factor and reorder
Is "current" selection valid ?
N
Y
Recompute "current" GDOP trace and add to list
Is "current" GDOP ≤ 1.15 "new" GDOP?
N
Y
Does "new" require station change?
Y
N
Replace "new" with next best choice from minimum trace list.
Is "new" GDOP ≤ "current" GDOP ?
N
Y
Use "current" selection
Use "new" selection
EXIT

Figure 42 Flow Diagram for Least Squares OMEGA Station Selection (Cont'd)

7.3 SAMPLE RESULTS

Often there is not one obvious best choice of LOP selection. Typically many different choices have very similar GDOPs, as is illustrated in the following example of a single application of the algorithm described in chapters 3 to 6.1 above.

In this example the location used is in the northeast Pacific, just off the coast of Vancouver Island, at latitude 48.5°N and longitude 126.°W. The date was Julian day 134 (May 14) and the time was 6:00 AM. There were assumed to be originally 8 acceptable stations (no operator or receiver deselections):

A, B, C, D, E, F, G, H

The S/N Ratios, provided in real time by the MX1105, were simulated for this test example somewhat arbitrarily, but using the charts in reference [10], as a rough guide to which stations should and should not be available. The MX1105 values (as described in section 3.1) for these 8 selected S/N ratios are as follows:

.11, .19, .85, .95, .05, .02, .35, .45

The station elimination software therefore rejected stations A,B,E and F because of low S/N Ratio (less than .2). Since this left only 4 acceptable stations, the software retested for S/N with a lower threshold (.1), thereby reselecting stations A and B. The modal interference routine then predicted that none of the stations would be subject to modal interference, leaving 6 stations from which to choose 4 LOPs:

A, B, C, D, G, H

Thus, as seen from table III, there were 6 ways of choosing 5 stations, and for each choice of stations there were 210 different choices of LOPs, which the simple linear dependence test reduced to 135 (10 of which are still not linearly independent). Figures 43 and 44 show the expected accuracies J (from equation (42)) for all 135 LOP choices for each of the 6 station selections. Here the 135 results from each station selection have been ordered and presented as one curve, identified by the station that was **not** used. From these figures we see that for each station selection about 10% of the LOP selections produce very bad geometry, and the remaining selections produce roughly equal and relatively good geometry. We can also see that in this case, since only one station must be eliminated, the station can be ordered from the best station D to the worst B.

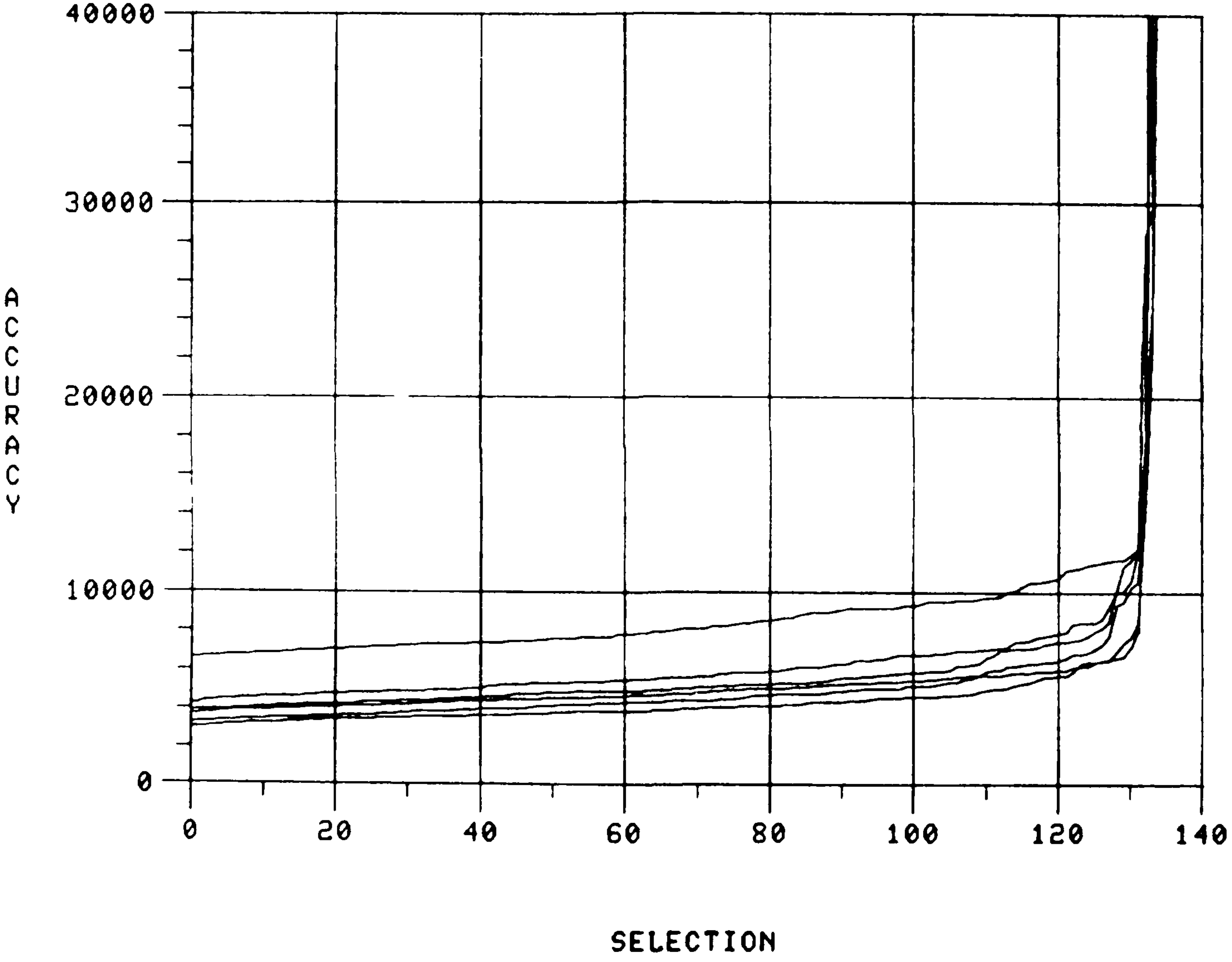
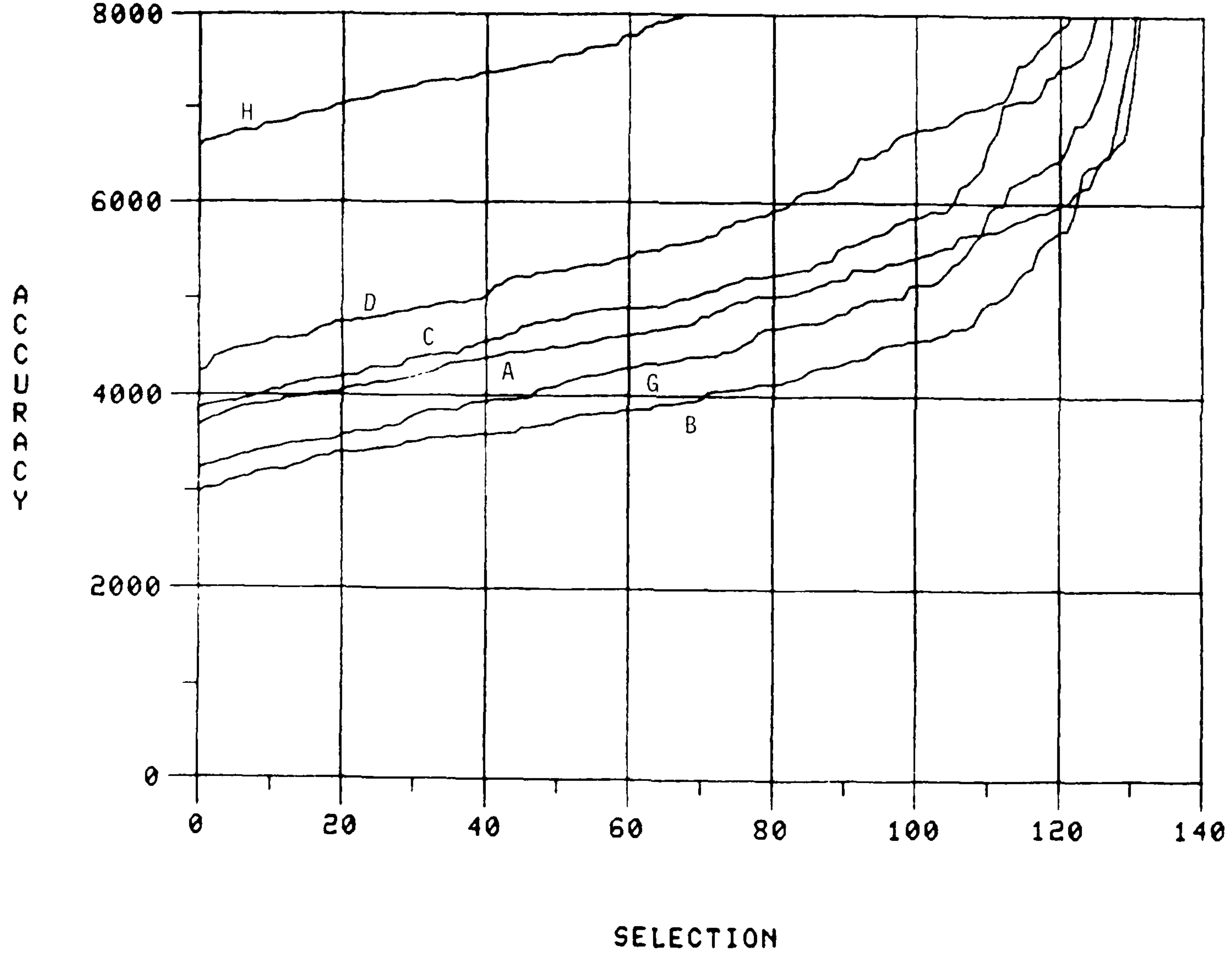


Figure 43. SAMPLE EXPECTED POSITION ACCURACIES

11-SEP-87

11-SEP-87

Figure 44. DETAIL OF SAMPLE EXPECTED POSITION ACCURACIES



From these "good geometry" selections, it is clear that the dominant factor in determining position accuracy is the station selection rather than the LOP selection. This fact could possibly be used to improve the computational efficiency of the selection algorithm, if a reliable method was found to eliminate the poor station selections without having to perform the GDOP calculation on all 135 LOP selections.

The preliminary selection routine chose the 8 best sets of LOPs, as shown in table IV below, with the indicated expected positional accuracy. Notice that, as would be expected by looking at figure 44, all 8 best sets of LOPs came from the same station selection, that which omitted station B.

Choices 2 and 3 in table IV were rejected at the end of the preselection stage because they were not linearly independent. The final selection stage boosted the expected error of the first, fourth, sixth and eighth choices by 20% (to 3,574., 3,633., 3,721. and 3,776. respectively) because in each case three of the LOPs use the same station (C in the first and eighth choices and D in the fourth and sixth). This results in the final best choice being the fifth preliminary choice:

C D	3,089.
C H	
D G	
G A	

Note that the expected error here is less than the next best choice by only the very narrowest of margins (about 1%) which is not at all significant, considering the statistical nature of the errors, and the rough approximation of equation (14). Therefore if we had a previous selection, which was the "best" selection one hour earlier, then it is quite likely that one of the preliminary selections here would use the same stations as the previous selection, and be sufficiently close to the "optimum" selection in expected accuracy. In that case a station change would not be necessary.

TABLE IV. Sample Results

CHOICE	LOPs	EXPECTED ERROR (metres)
1	C D C G C H G A	2,978.
2	C D C H D H G A	3,013.
3	C A D G D H G H	3,015.
4	C D D G D H H A	3,027.
5	C D C H D G G A	3,089.
6	C H D G D H D A	3,101.
7	C D C A D H G H	3,131.
8	C D C H C A G A	3,147.

8.0 SELECTION FOR KALMAN FILTER

Using the accuracy criterion J of equation (49) of section 6.2, for a Kalman filter, experiment has shown that once the 5 stations have been chosen the LOP selection does not have any effect at all on the expected accuracy (provided there is linear independence). Therefore it is not necessary to perform the full LOP enumeration, as was done for the least squares solution. Therefore the selection for the filter is done in one stage, and is much more computationally efficient.

Although the LOP selection doesn't affect the expected position accuracy, the station selection does, and a representative LOP selection must be made to evaluate each station selection. Also the LOP selection affects the availability of measurements in the event of signal loss. For example if all LOPs shared one station in common and the signal from that station for some reason became unuseable, then all LOPs would be lost. To avoid this problem, and to provide reasonable geometry for the least squares backup solution that MINS also provides, we try to make a reasonably good choice of LOPs. The simple method chosen is described in section 8.1 below.

The overall selection process is much the same as for the least squares case, as described in chapter 7, except that here:

- rather than enumerating and testing all LOPs, one representative LOP is explicitly constructed for testing.
- the selection criterion J is different.
- the reliability factor is enforced by the construction of each LOP.
- since each station selection has only one associated LOP, the efficiency measure (only changing selection if it leads to an improvement of more than 15%) is much simpler to apply.

The selection process for MINS is shown in figure 45, which appears similar to figure 42, but the absence of the LOP-generating loop here leads to much less computation and there is no need to search for a good LOP selection that doesn't involve a change of stations.

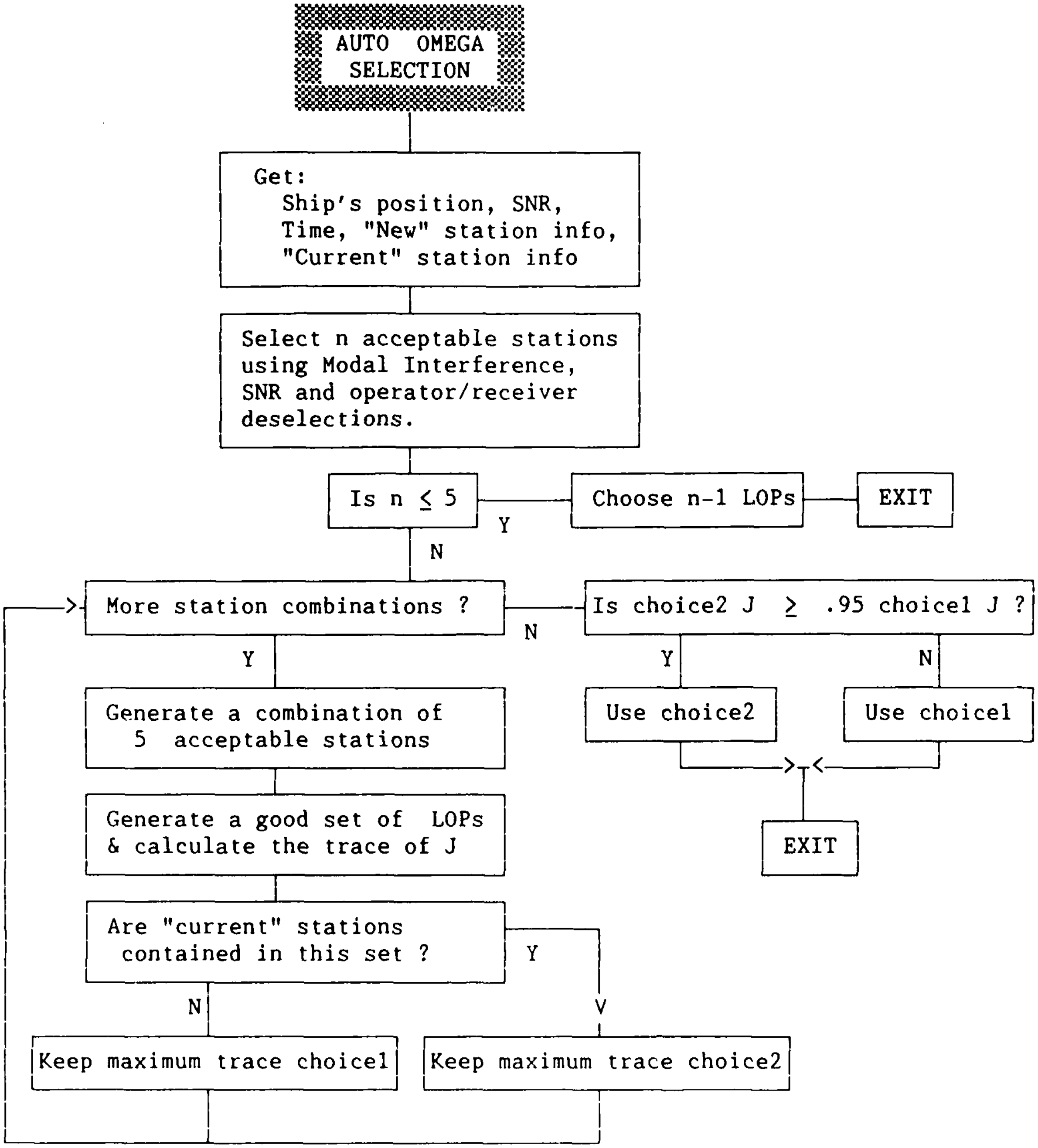

AUTO OMEGA SELECTION
Get:
Ship's position, SNR,
Time, "New" station info,
"Current" station info
Select n acceptable stations using Modal Interference, SNR and operator/receiver deselections.
Is n ≤ 5
Y
Choose n-1 LOPs
EXIT
N
More station combinations ?
N
Is choice2 J ≥ .95 choice1 J ?
Y
N
Use choice2
Use choice1
EXIT
Y
Generate a combination of 5 acceptable stations
Generate a good set of LOPs & calculate the trace of J
Are "current" stations contained in this set ?
Y
N
Keep maximum trace choice1
Keep maximum trace choice2

Figure 45 Flow Diagram for MINS OMEGA Station Selection

8.1 LOP SELECTION

When 6 or more stations are available after station elimination, then we must evaluate all possible sets of 5 stations (since the MINS Kalman filter can model the phase errors of at most 5 stations at a time). To do this we must be able to define a good set of 4 LOPs from any given 5 stations. If 5 or fewer stations are available then they will all be used and therefore no evaluation is necessary, however a reasonable choise of LOPs must still be formed. Here we describe how this can be done quite simply.

FIVE STATION CASE:

With 5 stations there are only 5|2 = 10 different LOPs, but this leads to 10|4 = 210 possible selections. It should be possible to greatly reduce this number by using the bearing angles to the 5 chosen transmitters Ψ, which are already available for the GDOP calculation, as described in chapter 6.0. For a generic solution it will be helpful to order the stations by this bearing angle, and label them accordingly:

$$S_1, \quad S_2, \quad S_3, \quad S_4, \quad S_5$$

so that

$$\Psi_1 < \Psi_2 < \Psi_3 < \Psi_4 < \Psi_5$$

One criteria for choosing LOPs is to pair stations that have widely separated bearing angles. By simply avoiding the pairing of consecutive stations (as ordered by bearing angle) the number of candidate LOPs is reduced from 10 to 5. This simple step greatly reduces the number of LOP choices from 210 to 5|4 = 5. These are:

$S_1 - S_3$

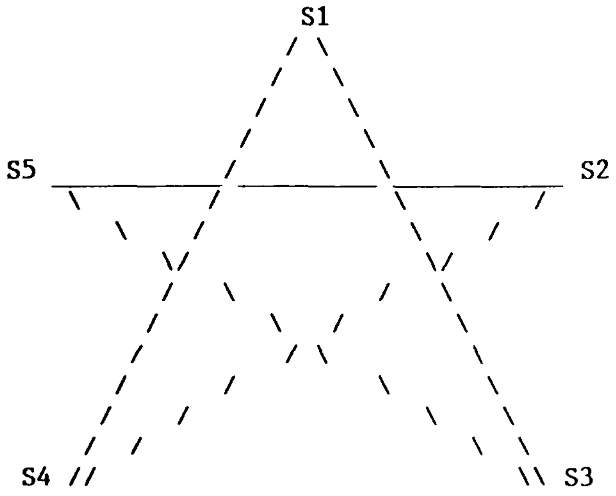
$S_3 \quad S_5$

$S_5 - S_2$

$S_2 - S_4$

$S_4 - S_1$

This is shown graphically below, where we can see the baselines of these 5 LOPs form a star. This leaves only one LOP to be eliminated. Another criteria for selection is to minimize the number of LOPs likely to be lost if a station is lost. Since the station most likely to be lost is the one with the lowest S/N ratio, a very simple way of doing this is to find the station with the lowest S/N ratio, S_i, and eliminate one of its LOPs, say S_i - S_j where j = (i-2)mod(5).



A more elaborate scheme could easily be devised to provide a better GDOP, but as was stated in section 8.0, this is really not necessary for the Kalman filter case.

FOUR STATION CASE:

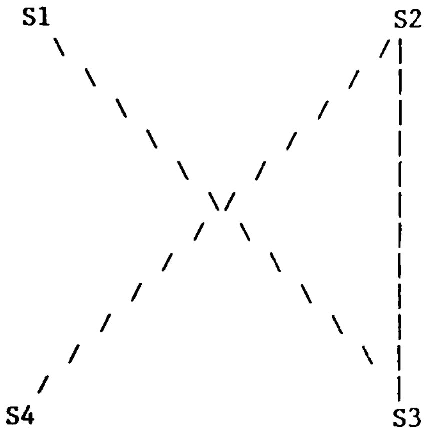
With only four stations to choose from we cannot avoid pairing adjacent stations. A simple choice could be obtained by ordering the stations by their bearing angles Ψ as described above but starting clockwise from the station with the lowest S/N ratio, and then selecting:

S1 - S3

S2 - S3

S2 - S4

which can be shown graphically as:



THREE STATION CASE:

When there are only three stations available the choise is very simple. One station must be paired with both remaining stations, so the common station should be chosen as the one with the highest S/N ratio.

8.2 SAMPLE RESULTS

When the example used in section 7.3 is reexamined using the Kalman filter selection algorithm, the station elimination process is exactly the same, resulting in the same 6 acceptable stations (ordered here according to their bearing angles):

A, B, D, C, G, H

with the following S/N ratio parameters:

.11, .19, .95, .85, .35, .45

which lead to the following expected phase error noise, modelled as $f(\rho)$, as given by equation (15):

3636., 2105., 421., 471., 1143., 889.

From these six stations there are six different subsets of 5 which must be evaluated based on the value of J, from equation (47). This depends on the phase noise covariance matrix **C**, which for each of these subsets is a 5x5 submatrix of the full covariance matrix:

$$\mathbf{C} = \begin{bmatrix} 13{,}220{,}496. & & & & & \\ & 4{,}431{,}025. & & & & \\ & & 177{,}241. & & & \\ & & & 221{,}841. & & \\ & & & & 1{,}306{,}449. & \\ & & & & & 790{,}321. \end{bmatrix}$$

The representative LOPs chosen, using the method described in section 8.1 above, are shown in table V, along with the value of $1/\sqrt{J}$, where J is the accuracy factor given by equation (49) of section 6.2. Although the selection algorithm actually maximizes J to determine the best choice, this is equivalent to minimizing $1/\sqrt{J}$, which we display here in table V because it is in units of metres and relates more directly to position error. In fact, comparing these numbers to the least squares position accuracies shown in figure 44, we see a strong similarity in the range of values.

From table V we can see that there are basically two poor choices (eliminating D or C) and 4 roughly equal choices. The order here is exactly according to the S/N ratio of the omitted station. This is not surprising since the criteria J centres on the covariance of the phase noise, which in this example has been given rather extreme values, as shown above.

The results here are not exactly the same as for the least-squares solution given in section 7.3 since of course the least squares criteria is centred on the covariance of the total phase error, which is less sensitive to S/N ratio. They are however similar in that they both indicate stations D, C and H to be more important, and stations A, B and G to be less important.

TABLE V. FILTER SAMPLE RESULTS

STATION OMITTED	LOPs	$1/\sqrt{J}$ "ERROR" (metres)
D	C H H B B G G A	662.
C	D H H B B G G A	418.
H	D G G B B C C A	309.
G	D H H B B C C A	295.
B	D G G A A C C H	289.
A	D G G B B C C H	287.

9.0 COMPUTATIONAL EFFICIENCY

By timing the entire enumeration and calculation of accuracy on the DREO/ED VAX 11-780, which we have also done with other MINS tasks, we can determine the relative computational burden of this automatic Omega station selection task. EDO has provided estimates of the load that the existing MINS tasks place on the real time MINS computer (68020 based), from which we can estimate what the loading of this Omega task would be on the 68020.

One area where significant optimisation was achieved is in the multiplication of the **D** matrix in equation (40) for the least squares case and (52) for the filter case. As seen in the computation count at the end of section 5.0, this is the only n^2 order term in the multiplication count. Fortunately **D** is a sparse matrix of ones and minus ones, and can be replaced by the use of the pointer array **P** (see equation (29)).

On the VAX 11-780, the full enumeration and selection of the 4 best LOPs from 7 stations (the worst case) for a least squares case requires 11.43 seconds. We didn't bother with the 8 station case because the target receiver MX1105 can only provide 7 pseudo-phase measurements, and in any case it is highly unlikely that all 8 signals will ever be acceptable at one time and place. The more likely situation will be the selection of 4 LOPs from 6 stations, which requires only 3.32 seconds. All other LOP selections require less than 2.1 seconds, as summarised in table VI below.

TABLE VI. Least Squares Selection Processing Burden

available stations	LOPs to choose	sets of stations	sets of n-1 LOPs	number of sets:		VAX CPU time taken
m	n-1	$m \mid n$		total	tested	(seconds)
7	4	21	210	4410	2835	11.43
6	4	6	210	1260	810	3.32
5	4	1	210	210	135	.63
7	3	35	20	700	560	2.06
6	3	15	20	300	240	1.06
5	3	5	20	100	80	.56
4	3	1	20	20	16	.32

The CPU times required to select stations for a Kalman filter are much smaller still, as shown in table VII.

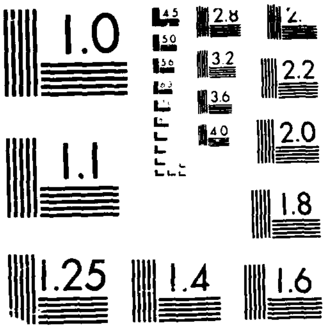
TABLE VII. Kalman filter selection Processing Burden

available stations	LOPs to choose	sets of stations	VAX CPU time taken
m	n-1	$m \mid n$	(seconds)
7	4	21	.30
6	4	6	.15
5	4	1	.07
4	3	1	.06

By comparison it takes about 39 seconds of cpu time for the MINS Navigation and Filter task to process 1.0 hours of data at a 1. minute filter rate on the VAX 11-780. The real-time system currently supports a filter rate of 30. seconds, implying that it is handling the equivalent of 78 seconds of "VAX time" per hour for this task. By changing the covariance propagation routine to a more efficient Bierman-Agee-Turner method (implemented in July 1987 at DREO), this 78 seconds per hour on the VAX has been reduced to 49 seconds, freeing the equivalent of 29 seconds of "VAX time". However we would like to increase the filter rate further from 30 seconds to 20. seconds, which will increase the filter processing burden by 50%, bringing it to 73.5 seconds of "VAX time" per hour of data. This is still a reduction of about 4.5 seconds per hour from what is currently being handled. From table VII it can be seen that this 4.5 seconds is more than enough time to perform the automatic Omega LOP selection once per hour.

AD-A193 302   AUTOMATIC OMEGA STATION AND LOP (LINE OF POSITION)   2/2
SELECTION(U) DEFENCE RESEARCH ESTABLISHMENT OTTAWA
(ONTARIO) J C MCMILLAN SEP 87 DREO-976

UNCLASSIFIED   F/G 17/7.2   NL



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS 1963 A

10.0 IMPLEMENTATION

The selection process described herin has been implemented at DREO in the "C" language on a VAX-780. The source code has been included here in Appendix A. To include this task into the realtime integrated navigation system MINS, requires additional software for interactions with certain other tasks and with the operator.

These additional details of implementation of the algorithms described above must necessarily be left to EDO Canada to include in MINS. Some of the details that they must address are listed here:

1/ modify the startup page
2/ modify the station selection page
3/ provide signal to noise ratio from the MX1105
4/ provide station deselection information from the MX1105 (if possible)
5/ synchronise with PPC task (timing controlled by EXEC)
6/ provide I/O to and from Navigation Task

11.0 ACKNOWLEDGEMENTS

Recognition and thanks are due to Eva Finklestein, Lum Hwang and Arthur Ficko for their help in the development, implementation and testing of the ideas described in this paper, and also to Peter Morris of the U.S. Department of Transportation for providing useful material.

REFERENCES

[1] J.C. McMillan, "A Kalman Filter for Marine Navigation", MPhil thesis, U. of Waterloo, 1980.

[2] D.F. Liang & J.C. McMillan, "Development Analysis of a Marine Integrated Navigation System", Proceeding of IEEE PLANS 82, December 1982.

[3] J.C. McMillan & A. Ficko, "Design and Implementation of the DREO Marine Integrated Navigation System (MINS) Volume II: Integration Algorithm and Software Development and MINS Software Listings", DREO report No. 886, September 1983.

[4] D.F. Liang, J.C. McMillan et el, "Low Cost Integrated Marine Navigation System", presented at the 39th Annual ION Meeting, Houston, June 1983, and published in Navigation, Journal of the Institute of Navigation, Vol. 30, No.4, Winter 1983-84.

[5] D.F. Liang, J.C. McMillan & C.A. Maskell, "Design and Test Evaluation of a Marine Integrated Navigation System", Proceedings of the Institute of Navigation National Technical Meeting, San Diego, January 1984.

[6] C.A. Maskell, "System Analysis and Sea-Trial Results of the DREO Marine Integrated Navigation System (MINS)", DREO Report No. 943, Feb. 1986.

[7] J.C. McMillan, "Optimal Compensation of Marine Navigation Sensor Errors", Proceedings of the 7th IFAC/IFORS Symposium on Identification & System Parameter Estimation, York, U.K., July 1985.

[8] "Navigator's Manual MX1105 Satellite/Omega Navigator", Magnavox Report R-5796E, Prepared by Magnavox Products and Systems Company 2829 Maricopa Street, Torrance, California, January 1982.

[9] P.B. Morris and R.R. Gupta, "Omega Navigational Signal Characteristics", Journal of the Institute of Navigation, Vol. 33, No. 3, Fall 1986.

[10] R.R. Gupta et el, "Omega Signal Coverage Prediction Diagrams for 10.2 kHz", TASC report for U.S. DOT, U.S. Coast Guard, ONSOD, October 1980.

[11] Admiralty Manual of Navigation, Volume III, 1955, London, Her Majesty's Stationery Office, revised 1973.

[12] M.I. Skolnik, "Introduction to Radar Systems", McGraw-Hill, 1962.

[13] Applied Optimal Estimation, edited by A. Gelb, M.I.T. Press, 1974.

APPENDIX A: COMPUTER PROGRAM

In this appendix we include a listing of the source code written in VAX "C" to perform the automatic Omega selection for MINS. Only the five main sets of modules specific to this function are listed. The mainline control routine is auto_omega.c, which calls the various subroutines of the other four modules to perform special functions such as to determine whether or not each signal will likely be subject to modal interference at the receiver location (in_MI_area.c), and whether or not the signal to noise ratio is acceptable (valid_snr.c). The source code modules are as follows:

```
auto_omega.c

        in_MI_area.c
                modal_inter_map.c
                day_light.c

        valid_snr.c

        compute_trace.c
                station_map.c
                station_change.c
                keep_stats.c

        combinations1.c
                enumerate_LOPs.c
```

The generic subroutines for matrix manipulation and geodetic calculations have been omitted. (Vincenty's method is used to calculate geodesic distances between locations and the associated bearing angles.)

```
#include "local.h"             /* standard include file */

/*
**----------------------------------------------------------------------------
** Project: MINS AUTO OMEGA STATION SELECTION
**
** Module name: auto_omega
**
** Description:
**
**      Automatic Omega station selection. Given the present available
**      stations, find the best LOPs using the following criteria:
**              - acceptable SNR
**              - not in Modal Interference Area
**              - good station geometry
**
** Usage:
**      auto_omega (latshp, lonshp, snr, day, gmt, new, old, stn_change,
**                  stn_op_disable)
**
** Input:
**      latshp  - latitude of ship (radians)
**      lonshp  - longitude of ship (radians)
**      snr     - array of signal-to-noise ratio for each OMEGA station.
**                (0. - 100.)
**      day     - day of the year (1 - 365 ???)
**      gmt     - time of the day (0.00 - 24.00)
**      new     - contains:
**                new.nlops  - User requested no. of lops
**                new.station_avail  -  array of BOOLEAN for each OMEGA station.

**      old     - contains:
**                old.nlops  - Current no. of lops
**                old.station  - array of current station selection (nlops+1)
**                old.pairs  - array of current lop pair set selection.
**      stn_op_disable - operator station disable array ( 0 - NSTNS)
**
** Output:
**      stn_change  -  This will tell us whether we need to change stations/lops
**                      or not. (BOOLEAN)
**
**      old     - contains: (depends on stn_change)
**                old.nlops  - the new/old no. of lops.
**                old.station  - array of new/old station selection.
**                old.pairs    - array of new/old lop pair set selection.
**                old.GDOP_trace  - GDOP calc.
**
**      IMPORTANT:
**              On exit, "old" will have all the information. Using
**              "stn_change", one can figure out whether "old" has
**              been updated or not.
```

```
**
**      NOTE:
**              old.nlops may be 0 on output (stn_change == TRUE) meaning
**              we don't have any good lops, so ignore OMEGA for now.
**
** REFERENCES:
**      "AUTOMATIC OMEGA STATION SELECTION", J.C. McMillan, DREO, 1987.
**
**---------------------------------------------------------------------------
*/

/*
** ---  USER INCLUDE SECTION & MACRO DEFINITION AREA
*/

#ifndef DEBUG
#define         DEBUG   0       /* 1- enable debug statements, 0 otherwise */
#endif

#ifndef  TEST                    /* 1- enable some print statements, 0 otherwise
*/
#define         TEST    0
#endif

#ifndef  TEST1                   /* 1- enable some print statements, 0 otherwise
*/
#define         TEST1   0
#endif

#ifndef  TEST2                   /* 1- enable some print statements, 0 otherwise
*/
#define         TEST2   1
#endif

#include "lopdef.h"             /* LOP defines */
#include "lopgbl.h"             /* LOP globals */

#define         MIN_KEEP                8
#define         SNR_THRESHOLD_1         0.2
#define         SNR_THRESHOLD_2         0.1
/*
** --------------------- GLOBAL DEFINITION AND EXTERNAL REFERENCE AREA --------
*/
struct  MINI
{
        double   min_trace;
        int      min_station[NSTNS];
        int      min_pair[NLOP][N];
        double   min_Mmat[N][N];
        int      pair_set_position;
};
```

```
FILE *binfile;
/*
** --- STATISTICAL VARIABLES
*/
        struct  MINI min_arr[MIN_KEEP+1];
        min_ptr[MAX_STATIONS+1];

        int     no_sets_per_station;
        int     total_no_sets;


/*
** ---  REFERENCES
*/
extern  char    station_map();


/*
** --------------------- MODULE USAGE AREA ---------------------------------
*/

auto_omega (latshp, lonshp, snr, day, gmt, new, old, stn_change, stn_op_disable)
int     day;
double  gmt;
double  latshp, lonshp;
double  snr[];
struct  CURRENT_SELECTION *old, *new;
BOOLEAN *stn_change;
BOOLEAN stn_op_disable[];

/*
** ---------------------  LOCAL DEFINITION AND REFERENCE AREA ----------------
*/
/*
** --- SYSTEM VARIABLES
*/
{
        register int     nstns;
        register int     nlops;
        int      station_choice[MAX_STATIONS];
        int      no_valid_stations;
        int      no_all_stations, a_station_selection[NSTNS];
        int      min_snr_stn;

        double   cur_GDOP_trace, min_snr;
        double   Mmat[N][N], R[MAX_STATIONS];


/*
** --- MISCELLANEOUS SCRATCH VARS
```

```
*/
        register int     i, j, k, l;
        int     station_occurrence;
        int     count1=0, count2=0, count3=0; /*BEWARE:DO NOT USE THESE VARS*/
        BOOLEAN same_station, valid_old_station;
        BOOLEAN prev_failed_snr[MAX_STATIONS];

/*
**  ------------------------- PROGRAM AREA -------------------------------
*/

/*
** -- Initialize range and bearing array needed for init_H matrix and
**    modal_inteference.
*/
        for (i=0; i<MAX_STATIONS; i++)
        {
            new->station_avail[i] &= !stn_op_disable[i];
            if (!new->station_avail[i]) continue;

            prev_failed_snr[i] = FALSE;
            geodaz(&latw[i], &longw[i], &latshp, &lonshp, &range_bearing[i][1],
                        &baz[i], &range_bearing[i][0]);

            range_bearing[i][1] /= 1000.;

            cos_H[i] = cos(baz[i]);
            sin_H[i] = sin(baz[i]);

            sin_stn_lat[i] = sin(latw[i]);
            sin_stn_lon[i] = sin(longw[i]);
            cos_stn_lat[i] = cos(latw[i]);
            cos_stn_lon[i] = cos(longw[i]);

            R[i] = (400.0)/snr[i];
        }

/*
** -- SELECT ACCEPTABLE STATIONS USING MODAL INTERFERENCE, SNR AND
** -- OPERATOR/RECEIVER DESELECTIONS.
*/
#if TEST2
        printf("\t PRESENT TIME (day, gmt): %d   %lf\n", day, gmt);
        printf("\t SHIP COORDINATES (lat, lon): %lf %lf\n", latshp*R2D,
                lonshp*R2D);
        printf("\t REQUESTED # OF LOPS : %d\n", new->nlops);
        printf("\t SNR : ");
        for (i=0; i < MAX_STATIONS; i++)
                if (new->station_avail[i])
                        printf(" %.2f ", snr[i]);
        printf("\n\n");
        printf("\t CURRENT NLOPS: %d\n", old->nlops);
        printf("\t CURRENT STATIONS : ");
        for (i=0; i < old->nlops+1; i++)
                printf(" %d(%c) ", old->station[i],
                        station_map(old->station[i]));
        printf("\n");
        printf("\t CURRENT PAIRS :\n");
        for (i=0; i < old->nlops; i++)
                printf("\t\t%d %d      %d %d      %c %c\n",
                        old->pair[i][0],old->pair[i][1],
                        old->station[old->pair[i][0]],
                        old->station[old->pair[i][1]],
                        station_map(old->station[old->pair[i][0]]),
                        station_map(old->station[old->pair[i][1]]));
        printf("\n\n");

        printf("\t NEW STATIONS : ");
        for (i=0; i < MAX_STATIONS; i++)
                if (new->station_avail[i])
                        printf(" %d(%c) ", i, station_map(i));
        printf("\n\n");
        printf("\t STATIONS DESELCTED BY M.I. OR BAD SNR:\n");
#endif

        no_valid_stations = 0;
        nlops = new->nlops;     /* User requested nlops */

        for (i=0; i < MAX_STATIONS; i++)
                if (new->station_avail[i])
                        if((prev_failed_snr[i] = !valid_snr(i, snr[i],
                                                SNR_THRESHOLD_1))
                            || in_MI_area(latshp, lonshp, i, day, gmt,
                                                &range_bearing[i]))
                                new->station_avail[i] = FALSE;
                        else
                                station_choice[no_valid_stations++] = i;

#if TEST2
        printf("\n\n");
        printf("\t STATIONS RESELCTED BY SNR:\n");
#endif

        /* Reselect SNR with lower threshold if not enough stations */
        if (no_valid_stations < 5)
                for (i=0; i < MAX_STATIONS; i++)
                        if (prev_failed_snr[i] && valid_snr(i, snr[i],
                                                        SNR_THRESHOLD_2))
                        {
                                new->station_avail[i] = TRUE;
                                station_choice[no_valid_stations++] = i;
#if TEST2
```

```
                                printf("\t\tSNR - reselected - %d(%c)\n",
                                        i, station_map(i));
#endif
                        }

        /* If more than 6 stations (inclusive), deselect the one with          */
        /* the highest.                                                         */

        if (no_valid_stations > 6)
        {
                min_snr = UNKNOWN_TRACE;        /* Any large # > 1.0            */
                for (i=0; i < MAX_STATIONS; i++)
                        if (new->station_avail[i] && snr[i] < min_snr)
                        {
                                min_snr = snr[i];
                                min_snr_stn = i;
                        }

                for (i=0; i < no_valid_stations; i++)
                        if (station_choice[i] == min_snr_stn)
                        {
#if TEST2
                                printf("\t\tSNR    -    min.    SNR   deselected    -
%d(%c)\n",
                                        i, station_map(i));
#endif
                                new->station_avail[i] = FALSE;
                                for (j=i+1; j < no_valid_stations; j++)
                                        station_choice[j-1] = station_choice[j];
                                no_valid_stations--;
                                break;
                        }
        }
/*
** -- INITIALIZE SYSTEM PARAMETERS FOR MINIMUM TRACE CALCULATIONS.
*/
        total_no_sets = 0;
        nstns  = MIN(nlops+1, no_valid_stations);
        nlops  = nstns-1;

        if (nlops <= 1) /** ?? CAN THIS EVER HAPPEN ???? ***/
        {
#if TEST
                printf("**** UNABLE TO CONTINUE\n");
                printf(" NSTNS = %d,   NLOPS = %d\n", nstns, nlops);
#endif
                *stn_change = TRUE;
                old->nlops = 0;
                return;
        }
/*
```

```
** ---  GENERATE COMBINATIONS FOR A STATION SELECTION (NSTNS <= 5) AND
**       SELECT A LOP PAIR.
*/
        /* Init stat. vars */
        for (i=0; i < MIN_KEEP; i++)
        {
                min_arr[i].min_trace = UNKNOWN_TRACE;
                min_ptr[i] = i;
        }


        while (combinations(no_valid_stations, nstns, &count2,
                &a_station_selection))
        {

                no_sets_per_station = 0;

#if TEST2
                printf("\n\n**************************************************\n");
                printf("\n\t\t NEW STATION :");

                binfile = fopen("station.out", "w");
#endif

                /* Set up a new station */
                for (j=0; j<nstns; j++)
                {
                        station[j] = station_choice[a_station_selection[j]];
#if TEST2
                        printf(" %d(%c) ", station[j], station_map(station[j]));
#endif
                }
#if TEST2
                printf("\n\n");
#endif

                /* Set up a set of NLOP pairs */
                count3 = 0;
                enumerate_LOPs(pair, station, nlops, R);
                no_sets_per_station++;
#if TEST1
                printf("-------------------------------------------------------
(%d)\n", total_no_sets+no_sets_per_station);
                printf("\n\t\t\t PAIRS \t\t STATIONS\n");

                for (j=0; j<nlops; j++)
                {
                        printf("\t\t\t %d %d\t\t %d %d \t\t %c %c\n",
                                pair[j][0],pair[j][1],
                                station[pair[j][0]], station[pair[j][1]],
                                station_map(station[pair[j][0]]),
                                station_map(station[pair[j][1]]));
                }
#endif

/*
** -- COMPUTE THE GDOP TRACE
*/
                compute_trace(&cur_GDOP_trace, Mmat, station, pair, nstns, R);
#if TEST2
                printf ("\n trace = %15.9lf %lf\n", cur_GDOP_trace,
                        1./sqrt(cur_GDOP_trace));
#endif
#if TEST1
                fprintf(binfile, "%016.9f\n", 1./sqrt(cur_GDOP_trace));
#endif

/*
** ---  KEEP STATISTICS FOR THE LAST MIN_KEEP (OR LESS) VALUES OF MIN. TRACE
**      The process is by keeping the mininum GDOP trace information in an
**      array of size MIN_KEEP in ascending order. A simple insertion sort
**      is used.
*/
                keep_stats(cur_GDOP_trace, nlops, Mmat);

                total_no_sets += no_sets_per_station;
#if TEST1
                fclose(binfile);
#endif
        }

/*
** --- PRINT OUT THE STATS AFTER EACH NEW STATION SELECTION
*/
#if TEST2
        print_stats(nlops);
#endif
/*
** ---  IF THE OLD STATION SELECTION IS VALID, WE TRY TO FIND THE BEST
**      STATION SELECTION BY MINIMIZING STATION CHANGES WITH VARIOUS
**      CONSTRAINTS.
*/

        /* Check if old selection is valid (i.e. old_station_avail is           */
        /*  a subset of the current station selection)                          */
        if (old->nlops != 0 && old->nlops <= nlops)
        {
                valid_old_station = TRUE;
                for (i=0; i < old->nlops+1; i++)
                        if (!new->station_avail[old->station[i]])
                        {
                                valid_old_station = FALSE;
                                break;
                        }
        }
        else  valid_old_station = FALSE;

#if TEST1
        if (valid_old_station)
        {
                printf("\n\t**** OLD STATION SELECTION IS VALID.\n");
                printf("\t**** Will process Old_GDOP_trace recalc\n");
        }
        else
        {
                printf("\n\t**** OLD STATION SELECTION IS NOT VALID.\n");
                printf("\t**** Returning the new selection \n");
        }
#endif

        /* If not a valid_old_station, we return the current selection. */
        /* At the same time saving "old".                                */
        if (!valid_old_station)
        {
                *stn_change = TRUE;
                station_change(&min_arr[min_ptr[0]], nlops, old);
                return;
        }

        /* Recompute the trace for the old station.                      */
        compute_trace(&old->GDOP_trace,     Mmat,     old->station,     old->pair,
old->nlops+1, R);

#if TEST1
        printf("\n\t OLD_GDOP_trace = %lf\n", old->GDOP_trace);
#endif


        /* Test the old trace against the best of "min_keep" new traces */
        /* and see if the new trace  is 15% or better.                   */
        if (old->GDOP_trace > (0.85 * min_arr[min_ptr[0]].min_trace))
        {
                /* Yes 15% better, so we use the new station */
#if TEST1
                printf("\t\t New station selection is 15%% better.\n");
#endif
                *stn_change = TRUE;
                station_change(&min_arr[min_ptr[0]], nlops, old);
                return;
        }
        else
        {
#if TEST1
```

```
                printf("\t\t We have to find closest match ... \n");
#endif
                /* Otherwise */
                same_station = FALSE;
                for (i=0; i < MIN_KEEP &&
                        min_arr[min_ptr[i]].min_trace < old->GDOP_trace; i++)
                {
                        /*  Set up flag to see if the old station selection is a
subset  */
                        /*          of          the          "best"          new
selection.                                      */
                        for (j=0; j < old->nlops+1; j++)
                        {
                                for (k=j; k < nstns; k++)
                                        if ((same_station=
                                                (min_arr[min_ptr[i]].min_station
[k] == old->station[j])))
                                                break;

                                if (same_station)
                                {
#if TEST1
                                        printf("\t\t old is subset of new
                                                station selection\n");
                                        printf("\t\t Change lops(stations).\n");
#endif
                                        *stn_change = TRUE;
                                        station_change(&min_arr[min_ptr[i]],
                                                        nlops, old);
                                        return;
                                }
                                else
                                        break;
                        }
                }
        }

#if TEST1
        printf("\t\t Don't change station .....\n");
#endif
        *stn_change = FALSE;

}
```

```
/******************************************************************/
/*       MINS AUTOMATIC OMEGA STATION SELECTION                   */
/*                                                                */
/* Routine: compute_trace()                                       */
/*                                                                */
/* Description:                                                   */
/*     Used to calculate the GDOP trace given station selection */
/*     information.                                               */
/*     The term "station selection" includes both station no.    */
/*     and LOP pair information.                                  */
/*                                                                */
/* Usage:                                                         */
/*      compute_trace(trace, Mmat, station, pair, nstns, R)      */
/*      double  *trace, Mmat[N][N];                               */
/*      int     station[], nstns;                                 */
/*      int     pair[][N];                                        */
/*      double  R[];                                              */
/*                                                                */
/*   Where:                                                       */
/*      trace   - GDOP trace of station selection.               */
/*      Mmat    - the 2x2 matrix whereby the trace is            */
/*                computed.                                       */
/*      station - An array (nstns) of valid Omega stations.      */
/*      nstns   - the no. of valid Omega stations for "station" */
/*                array.                                          */
/*      pair    - An array (nlops x 2) giving the LOP selection */
/*      R       - An array (nstns) giving SNR for the Omega      */
/*                stations.                                       */
/*                                                                */
/* Inputs:                                                        */
/*      station                                                   */
/*      nstns                                                     */
/*      pair                                                      */
/*      R                                                         */
/*                                                                */
/* Outputs:                                                       */
/*      Mmat                                                      */
/*      trace                                                     */
/*                                                                */
/* Returns:                                                       */
/*      none                                                      */
/*                                                                */
/* References:                                                    */
/*      "AUTOMATIC OMEGA STATION SELECTION", J.C. McMillan,      */
/*      1987.                                                     */
/*                                                                */
/******************************************************************/
compute_trace(trace, Mmat, station, pair, nstns, R)
double  *trace, Mmat[N][N];
int     station[], nstns;
int     pair[][N];
double  R[];
{
        /* System vars */
        register int     nlops = nstns -1;
        double   tH[N][NLOP], H2[N][N], invH[N][N];
        double   Hinv[N][NLOP], r2[N][NSTNS];
        double   tresult[NSTNS][N], H[NLOP][N];
        double   detH;

        double   result[N][NSTNS];
        double   D[NLOP][NSTNS];
        double   Dt[NSTNS][NLOP];

        double   Dresult[NLOP][NLOP];
        double   check[NLOP][NLOP];
        double   Dinv[NLOP][NLOP], Hinter[N][NLOP];

        /* Scratch vars */
        register int    i,j,k;



/*
** -- INITIALIZE WORK ARRAYS
*/
        for (j=0; j < NSTNS; j++)
                result[0][j] = result[1][j] = 0.;

        for (i=0; i < NLOP; i++)
                for (j=0; j < NSTNS; j++)
                {
                        D[i][j] = 0.;
                        if (j < NLOP)
                                Dresult[i][j] = 0.;
                }


        /* Init H matrix */
        init_Hmat(H, nlops, N, N, station, pair);


/*
** ---  CALCULATE  trace (H * inverse(D * R * Dt) * Ht)
*/

#if TEST2
        printf("\t H matrix is as follows: \n");
        printmat(H, nlops, N, N);
#endif
        /* tH =  H^T */
        transpose(H, tH, nlops, N, N, NLOP);

#if TEST2
        printf("\t tH matrix is as follows: \n");
        printmat(tH, N, nlops, NLOP);
#endif

        /*  D = D * sqrt(R) matrix */
        for (i=0; i < nlops; i++)
        {
                D[i][pair[i][0]] =  R[station[pair[i][0]]];
                D[i][pair[i][1]] = -R[station[pair[i][1]]];
        }
#if TEST2
        printf("\t D matrix is as follows: \n");
        printmat(D, nlops, nstns, NSTNS);
#endif
        /*  Dresult = D * sqrt(R) * transpose (D * sqrt(R)) */
        for (i=0; i < nlops; i++)
                for (j=0; j < nlops; j++)
                        for (k=0; k < nstns; k++)
                                Dresult[i][j] += D[i][k] * D[j][k];
#if TEST2
        printf("\t D*R*Dt matrix is as follows: \n");
        printmat(Dresult, nlops, nlops, NLOP);
#endif

        /* Dinv = Inverse(Dresult) */
        matinv(Dresult, nlops, Dinv );
#if TEST2
        printf("\t Inverse(D*R*Dt) matrix is as follows: \n");
        printmat(Dinv, nlops, nlops, NLOP);
#endif

        /* Hinter = Ht * Dresult */
        mult(tH, Dinv, Hinter, N, nlops, nlops, NLOP, NLOP);
#if TEST2
        printf("\t Ht * Inverse(D*R*Dt) matrix is as follows: \n");
        printmat(Hinter, N, nlops, NLOP);
#endif

        /* Mmat = Hinter * H */
        mult(Hinter, H, Mmat, N, nlops, N, NLOP, N);

        /* trace = trace(Mmat) */
        *trace = Mmat[0][0] + Mmat[1][1];

}
```

```
/****************************************************************/
/*       MINS AUTOMATIC OMEGA STATION SELECTION                 */
/*                                                              */
/* Function: station_map()                                      */
/*                                                              */
/* Description:                                                 */
/*      Maps a station # into it's corresponding                */
/*      station character.                                      */
/*                                                              */
/*                                                              */
/* Usage:                                                       */
/*      char station_map(station)                               */
/*                                                              */
/*    Where:                                                    */
/*      station - OMEGA station id (0-7)                        */
/*                                                              */
/* Inputs:                                                      */
/*      station                                                 */
/* Outputs:                                                     */
/*                                                              */
/* Returns:                                                     */
/*      returns a character 'A' to 'H' corresponding to station */
/*      (0-7) and '?' if the station id is invalid.             */
/*                                                              */
/****************************************************************/
char station_map(station)
{
        return((station >= 0 && station < MAX_STATIONS) ?
                        (char)station+(int)'A' : '?');
}

/****************************************************************/
/*       MINS AUTOMATIC OMEGA STATION SELECTION                 */
/*                                                              */
/* Routine: station_change()                                    */
/*                                                              */
/* Description:                                                 */
/*      Updates the CURRENT_SELECTION structure, "old" which    */
/*      holds information about the "current" station selection */
/*      (ie. either station or LOP pair change).                */
/*                                                              */
/*      "returnee" contains the new OMEGA station selection.    */
/*                                                              */
/* Usage:                                                       */
/*      station_change(returnee, nlops, old)                    */
/*      struct MINI *returnee;                                  */
/*      struct CURRENT_SELECTION  *old;                         */
/*      register int    nlops;                                  */
/*                                                              */
/*   Where:                                                     */
/*      returnee        - structure containing new OMEGA        */
/*                        station selection.                    */
/*      nlops           - the new no. of LOPs. (because          */
/*                        struct MINI does not have this)       */
/*      old             - the "current" station selection.      */
/*                                                              */
/* Inputs:                                                      */
/*      returnee, nlops                                         */
/*                                                              */
/* Outputs:                                                     */
/*      old                                                     */
/*                                                              */
/* Returns:                                                     */
/*                                                              */
/****************************************************************/

station_change(returnee, nlops, old)
struct MINI *returnee;
struct CURRENT_SELECTION  *old;
register int    nlops;
{
        int i;


        old->nlops = nlops;

        for (i=0; i < nlops; i++)
        {
                old->pair[i][0] = returnee->min_pair[i][0];
                old->pair[i][1] = returnee->min_pair[i][1];
                old->station[i] = returnee->min_station[i];
        }
        /* Note: nstns = nlops + 1 = i */
        old->station[i] = returnee->min_station[i];

        old->GDOP_trace = returnee->min_trace;
}

/****************************************************************/
/*       MINS AUTOMATIC OMEGA STATION SELECTION                 */
/*                                                              */
/* Routine: keep_stats()                                        */
/*                                                              */
/* Description:                                                 */
/*      Keeps track of an array of "min_keep" minimum traces.  */
/*      Initially, all traces in the array is initialized to   */
/*      UNKNOWN_TRACE (99999.).                                 */
/*                                                              */
/*      Stats are kept in ascending order by an index array,   */
/*      min_ptr[].                                              */
/*                                                              */
/* Usage:                                                       */
/*      keep_stats(cur_GDOP_trace, nlops, Mmat)                           */
/*      double  cur_GDOP_trace;                                           */
/*      int     nlops;                                                    */
/*      double  Mmat[N][N];                                               */
/*                                                                        */
/*    Where:                                                              */
/*      cur_GDOP_trace  - a trace to be compared and possibly             */
/*                        included in the minimum trace list.             */
/*      nlops           - the number of LOPs                              */
/*      Mmat            - Array from which trace is calculated            */
/*                                                                        */
/* Inputs:                                                                */
/*      all of parameter list                                             */
/*                                                                        */
/* Outputs:                                                               */
/*      none                                                              */
/*                                                                        */
/* Returns:                                                               */
/*                                                                        */
/*************************************************************************/

keep_stats(cur_GDOP_trace, nlops, Mmat)
double  cur_GDOP_trace;
int     nlops;
double  Mmat[N][N];
{
        register int i, j, k;
        BOOLEAN found;

        /* MIN_KEEP is the number of values to keep (0-7) */

        /* Compare the current GDOP trace with the list and stop at the first */
        for (i=MIN_KEEP; i > 0 &&
                (cur_GDOP_trace < min_arr[min_ptr[i-1]].min_trace); i--)
                min_ptr[i] = min_ptr[i-1];


        if (i != MIN_KEEP)      /* Yes, we need to insert this value */
        {
                min_ptr[i] = min_ptr[MIN_KEEP];

                /* Now insert new selection into slot i */
                for (j=0; j < nlops; j++)
                {
                        min_arr[min_ptr[i]].min_pair[j][0] = pair[j][0];
                        min_arr[min_ptr[i]].min_pair[j][1] = pair[j][1];
                }

                for (j=0; j < nlops+1; j++)
                        min_arr[min_ptr[i]].min_station[j] = station[j];
                for (j=0; j < N; j++)
                {
                        min_arr[min_ptr[i]].min_Mmat[j][0] = Mmat[j][0];
                        min_arr[min_ptr[i]].min_Mmat[j][1] = Mmat[j][1];
                }

                min_arr[min_ptr[i]].min_trace = cur_GDOP_trace;
                min_arr[min_ptr[i]].pair_set_position    =    total_no_sets    +
no_sets_per_station;
        }
}


/****************************************************************/
/*       MINS AUTOMATIC OMEGA STATION SELECTION                 */
/*                                                              */
/* Routine: print_stats()                                       */
/*                                                              */
/* Description:                                                 */
/*      Used to print statistics kept by minimum trace list.    */
/*                                                              */
/* Usage:                                                       */
/*      print_stats(nlops)                                      */
/*      int nlops;                                              */
/*                                                              */
/*    Where:                                                    */
/*      nlops   - no. of LOPs. (nstns = nlops + 1)              */
/*                                                              */
/* Inputs:                                                      */
/*      nlops                                                   */
/*                                                              */
/* Outputs:                                                     */
/*                                                              */
/* Returns:                                                     */
/*                                                              */
/****************************************************************/

print_stats(nlops)
int nlops;
{
        register int i, j, k;

#if TEST2
        printf("\n\n######################################################\n");
        printf("\n\t\t PAIR STATS  FOR LAST %d (OR LESS):\n\n", MIN_KEEP);

        for (i=0; i< MIN_KEEP && min_arr[min_ptr[i]].min_trace != UNKNOWN_TRACE;
i++)
        {
                for (j=0; j<nlops; j++)
                {
                        printf("\t\t\t %d %d\t\t %d %d \t\t %c %c\n",
                                min_arr[min_ptr[i]].min_pair[j][0],min_arr[min_p
tr[i]].min_pair[j][1],
                                min_arr[min_ptr[i]].min_station[min_arr[min_ptr[
i]].min_pair[j][0]],
                                min_arr[min_ptr[i]].min_station[min_arr[min_ptr[
i]].min_pair[j][1]],
                                station_map(min_arr[min_ptr[i]].min_station[min_
arr[min_ptr[i]].min_pair[j][0]]),
                                station_map(min_arr[min_ptr[i]].min_station[min_
arr[min_ptr[i]].min_pair[j][1]]));
                }
                printf("\n\t\t\t MMAT:\n");
                for (j=0; j<N; j++)
                {
                        printf("\t\t\t\t %.8f %.8f\n",
                                min_arr[min_ptr[i]].min_Mmat[j][0],
min_arr[min_ptr[i]].min_Mmat[j][1]);
                }
                printf("\n\t\t       MINIMUM       TRACE       (%d):       %f",
min_arr[min_ptr[i]].pair_set_position,
                        min_arr[min_ptr[i]].min_trace);
                printf("\n\n");
        }

        printf("\n\t\t    NO.   OF   SETS   FOR   THIS   STATION   SELECTION:  %d\n",
no_sets_per_station);
        printf("\t\t TOTAL NO. OF SETS: %d\n", total_no_sets);
#endif
#if     TEST
        printf("\f\n\n\n");
#endif
}
```

```
#include "local.h"

/*****************************************************************/
/* Routine: in_MI_area()                                         */
/*                                                               */
/* Description:                                                  */
/*      Provided with a lat/lon position, Omega station, time    */
/*      and day-of-year, this routine will tell the caller       */
/*      whether the geographical position is in a modal          */
/*      Interference Area.                                       */
/*                                                               */
/*                                                               */
/* Usage:                                                        */
/*      in_MI_area(shplat, shplon, station_no, day, gmt,         */
/*                      range_bearing)                           */
/*        double gmt;                                            */
/*        int   station_no, day;                                 */
/*        double range_bearing[2];                               */
/*        double shplat, shplon;                                 */
/*                                                               */
/* Where:                                                        */
/*      shplat          - latitude of ship (radians)             */
/*      shplon          - longitude of ship (radians)            */
/*      station_no      - OMEGA station id.                      */
/*      day             - day (0-365th day)                      */
/*      gmt             - time of day (0. - 24. hours)           */
/*      range_bearing   - modal interference function map        */
/*                        (hard-coded into gen_modal_inter.h)    */
/* Inputs:                                                       */
/*      all parameters in parameter list.                        */
/*                                                               */
/* Outputs:                                                      */
/*      none                                                     */
/*                                                               */
/* Returns:                                                      */
/*      TRUE - ship is in modal inteference area                 */
/*      FALSE - otherwise                                        */
/*                                                               */
/* References:                                                   */
/*      ONSOD (vol. III) for modal interference maps.            */
/*                                                               */
/*****************************************************************/

#include "lopdef.h"
#include "lopext.h"
#include "gen_modal_inter.h"
#define  TEST  1
#define  DEBUG 0

/* Near-field and far-field constraints */
#define         FUNCTION_MAX    19000.  /* Km */
```

```
#define         FUNCTION_MIN    500.    /* Km */


#define         MODAL_INTER_MIN(X)      (X)
#define         MODAL_INTER_MAX(X)      (X)+MAX_STATIONS

extern BOOLEAN day_light();


BOOLEAN in_MI_area(shplat, shplon, station_no, day, gmt, range_bearing)
double gmt;
register int    station_no, day;
double range_bearing[2];
double shplat, shplon;
{
        double range_function;

/*
** -- Check against min and max consts, for near-field and wrong-way-path areas
*/
        if (range_bearing[1] > FUNCTION_MAX)
        {
#if TEST
        printf("\t\tM.I. - Greater than %d    STN= %d(%c)\n", FUNCTION_MAX,
                                station_no, station_map(station_no));
#endif
                return(TRUE);
        }

        if (range_bearing[1] < FUNCTION_MIN)
        {
#if TEST
        printf("\t\tM.I. - Less than %d       STN=%d(%c)\n", FUNCTION_MIN,
                                station_no, station_map(station_no));
#endif
                return(TRUE);
        }

/*
** -- Check if the path more or less than 1/3 in darkness,
**    If it is less, then assume no modal interference.
*/
        if (day_light(gmt, day, shplat, shplon, station_no))   /* < 1/3 dark? */
                return(FALSE);  /* No modal inteference */

/*
** -- Check if the range at the current bearing is between the MAX and MIN
**    functions of the modal interference map of the station.
**    (we already know that it is between 19000. and 500. Km
*/
```

```
        /* generate the MAX-function range at this bearing */
        modal_inter_map(MODAL_INTER_MAX(station_no), range_bearing[0], &range_function);

        if (range_bearing[1] >= range_function)
        {
#if TEST
        printf("\t\tM.I. - In MAX function    STN = %d(%c)\n",
                        station_no, station_map(station_no));
#endif
                return(TRUE);   /* We're in Modal interference zone */
        }

        /* generate the MIN-function range at this bearing */
        modal_inter_map(MODAL_INTER_MIN(station_no), range_bearing[0], &range_function);

        if  (range_bearing[1] <= range_function)
        {
#if TEST
        printf("\t\tM.I. - In MIN function    STN = %d(%c)\n",
                        station_no, station_map(station_no));
#endif
                return(TRUE);   /* We're in Modal interference zone */
        }

        /* Hey, we're not in the modal interference area */
        return(FALSE);
}


/*****************************************************************/
/* Routine: modal_inter_map()                                    */
/*                                                               */
/* Description:                                                  */
/*      Given the station id and the bearing of the ship         */
/*      to the OMEGA station, the routine will calculate         */
/*      the range of either the far/near field of the modal      */
/*      interference map function.                               */
/*                                                               */
/*      Modal inteference map functions are implemented as       */
/*      Fourier tranforms and then reduced to it's most          */
/*      efficient form.                                          */
/*                                                               */
/* Usage:                                                        */
/*      modal_inter_map(station_no, theta, range)                */
/*       int    station_no;                                      */
/*       double theta;                                           */
/*       double *range;                                          */
/*                                                               */
/* where:                                                        */
/*      station_no      - Omega station id. and function         */
/*                        0-7 represents near-field              */
/*                         8-15 represents far-field          */
/*      theta            - bearing of the ship from the station */
/*                         (radians)                           */
/*      range            - range of the ship from the station   */
/*                         (Km)                                */
/*                                                             */
/* Input vars:                                                 */
/*      station_no, theta                                      */
/*                                                             */
/* Output vars:                                                */
/*      range                                                  */
/*                                                             */
/* References:                                                 */
/*      N/A                                                    */
/*                                                             */
/*****************************************************************/

modal_inter_map(station_no, theta, range)
register int    station_no;
double theta;
double *range;
{

        double c;
        double x, x2, x3, x5;
        double y, y2, y3, y4, y5, y6;
        double xy, xy3, xy5, x3y;

        x       = sin(theta);
        y       = cos(theta);
        xy      = x  * y;
        x2      = x  * x;
        y2      = y  * y;
        x3      = x2 * x;
        y3      = y2 * y;

        y4      = y2 * y2;
        x5      = x2 * x3;
        y5      = y2 * y3;
        y6      = y3 * y3;
        xy3     = x  * y3;
        xy5     = x  * y5;
        x3y     = x3 * y;

        *range =  f[station_no].c +
             xy * f[station_no].xy_coef + x3y * f[station_no].x3y_coef +
             xy3 * f[station_no].xy3_coef + xy5 * f[station_no].xy5_coef +
             x  * f[station_no].x_coef  + y   * f[station_no].y_coef +
             y2  * f[station_no].y2_coef    +
             x3 * f[station_no].x3_coef + y3  * f[station_no].y3_coef +
             y4  * f[station_no].y4_coef   + x5 * f[station_no].x5_coef +
             y5  * f[station_no].y5_coef + y6  * f[station_no].y6_coef;

}


/*****************************************************************/
/* Routine: modal_inter_map()                                   */
/*                                                              */
/* Description:                                                 */
/*      Is ship in day light or in darkness with respect        */
/*      to an OMEGA station (weighted).                         */
/*                                                              */
/*      This is done by taking the dot products of the planes  */
/*        generated between:                                    */
/*              - ship and center of earth.                     */
/*              - subpoint (weighted position) and center       */
/*                of earth.                                     */
/*              - center of sun and center of the earth.        */
/*                                                              */
/* Usage:                                                       */
/*      BOOLEAN day_light(gmt, day, shplat, shplon, stn)        */
/*      double    gmt, shplat, shplon;                          */
/*      int       day, stn;                                     */
/*                                                              */
/* where:                                                       */
/*      gmt             - Time of day (0. - 24. hours)          */
/*      day             - day of year (0-356)                   */
/*      shplat          - latitude of ship (radians)            */
/*      shplon          - longitude of ship (radians)           */
/*      stn             - OMEGA station ID.                     */
/*                                                              */
/* Input vars:                                                  */
/*      all parameters specified.                               */
/*                                                              */
/* Output vars:                                                 */
/*      none                                                    */
/*                                                              */
/* Returns:                                                     */
/*      TRUE  - If in weighted day light.                       */
/*      FALSE - otherwise.                                      */
/*                                                              */
/* References:                                                  */
/*      N/A                                                     */
/*                                                              */
/*****************************************************************/

#define  EARTH_INCLINATION  0.409126121         /* 23.4412 deg */

BOOLEAN day_light(gmt, day, shplat, shplon, stn)
double    gmt, shplat, shplon;
```

```
register int       day, stn;
{

   double   wghtshp[2] = {.5, 2.};               /* WEIGHTS */
   double   sublat, sublon, scalar_product[2];
   double   subvec[3], shpvec[3], stnvec[3], x[3];
   register int       i, j;

   sublat   = EARTH_INCLINATION * sin(TWOPI * (day-80.)/365.);

   sublon   = (12.-gmt)/11.967222 * PI;


   if (sublon > PI)
       sublon = sublon - TWOPI;
   else if (sublon < -PI)
       sublon = sublon + TWOPI;

#if DEBUG
   printf("\n SUBPOINT LATITUDE:  %f\n", sublat);
   printf("\n SUBPOINT LONGITUDE: %f\n", sublon);
#endif

/*
** --- Form the unit vector to the solar subpoint, the ship's location
**     and the Omega station.
*/

   subvec[0] = sin(sublat);
   subvec[1] = sin(sublon) * cos(sublat);
   subvec[2] = cos(sublon) * cos(sublat);

   shpvec[0] = sin(shplat);
   shpvec[1] = sin(shplon) * cos(shplat);
   shpvec[2] = cos(shplon) * cos(shplat);

   stnvec[0] = sin_stn_lat[stn];
   stnvec[1] = sin_stn_lon[stn] * cos_stn_lat[stn];
   stnvec[2] = cos_stn_lon[stn] * cos_stn_lat[stn];

/*
** --- Form dot product of solar subpoint vector with vectors pointing
**     to locations on Omega signal path 1/3 distance from each end.
*/

   for (j=0; j<2; j++)
   {
        scalar_product[j] = 0.;
        for (i=0; i<3; i++)
                scalar_product[j] +=
                        (wghtshp[j] * shpvec[i] + stnvec[i]) * subvec[i];
   }

/*
** --- If both projections are positive, then signal path is more than
**     2/3 in daylight.
*/
   return(((scalar_product[0] > 0. && scalar_product[1] > 0.) ? TRUE : FALSE));
}
```

```
#include "local.h"

/******************************************************************/
/*        MINS AUTOMATIC OMEGA STATION SELECTION                  */
/*                                                                */
/* Routine: valid_snr                                             */
/*                                                                */
/* Description:                                                   */
/*      Function checks to see if the Signal-to-Noise-Ratio       */
/*      is valid.                                                 */
/*                                                                */
/* Usage:                                                         */
/*      BOOLEAN valid_snr(station_no, snr, threshold)             */
/*      int     station_no;                                       */
/*      double  snr;                                              */
/*      float   threshold;                                        */
/*                                                                */
/* Where:                                                         */
/*      station_no      - the OMEGA station id. (0-7)             */
/*      snr             - the present SNR for the station         */
/*                                                                */
/* Inputs vars:                                                   */
/*      all parameters                                            */
/*                                                                */
/* Outputs vars:                                                  */
/*      none                                                      */
/*                                                                */
/* Returns:                                                       */
/*      TRUE - if good SNR, else FALSE                            */
/*                                                                */
/* References:                                                    */
/*                                                                */
/******************************************************************/

#define TEST 1

BOOLEAN valid_snr(station_no, snr, threshold)
int     station_no;
double  snr;
float   threshold;
{

#if TEST
        if (snr <= threshold)
        printf("\t\tSNR  - Rejected          STN = %d(%c)\n", station_no, station_map
(station_no));
#endif
        return((snr <= threshold) ? FALSE : TRUE);
}
```

```
#include "local.h"
#include "lopdef.h"
#include "lopext.h"

/************************************************************************/
/*                                                                      */
/* Function: combinations                                               */
/*                                                                      */
/* Description:                                                         */
/*      This function will generate n|r ("n" choose "r")                */
/*      combinations of numbers (range 0:n-1)                           */
/*                                                                      */
/*      When all combinations are generated, the function will          */
/*      return 0.                                                       */
/*                                                                      */
/*      On initialization, variable "total_comb" must be 0.             */
/*                                                                      */
/* Syntax:                                                              */
/*      combinations(n, r, total_comb, result)                          */
/*                                                                      */
/*    where:                                                            */
/*      int n;          "n" numbers                                     */
/*      int r;          sets of "r" size combinations                   */
/*      int *total_comb;        used as a counter.                      */
/*                      This value must be 0 at the first call          */
/*                      to the function.                                */
/*                      This value MUST ONLY be modified by             */
/*                      the function.                                   */
/*      int result[];   Array of at least "r" length. Used to store     */
/*                      the resulting combination generated.            */
/*                                                                      */
/* Returns:                                                             */
/*       0  - when all combinations have been generated.                */
/*       1  - otherwise.                                                */
/*                                                                      */
/************************************************************************/

int combinations(n, r, total_comb, result)
register int n, r;
int *total_comb;
int *result;
{
        register int i, j, k;


        if (*total_comb <= 0) /* Initialization */
        {
                (*total_comb)++;
                for (i=0; i<r; i++)
                        result[i] = i;
                return(1);
        }
        else
        {       /* Set up a combination */
                (*total_comb)++;
                j = r-1;
                while (j >= 0)
                {
                        k = n-(r-j);
                        if (result[j] < k)
                        {
                                result[j]++;
                                for (i = j+1; i < r; i++)
                                        result[i] = result[i-1] + 1;
                                return(1);
                        }
                        else
                                j--;
                }

        }
        return(0);
}
/********************************************************************/

#ifdef DEBUG
#undef DEBUG
#endif
#define DEBUG 1

enumerate_LOPs(pairs, stations, nlops, R)
int     pairs[][N];
int     stations[];
int     nlops;
double  R[];
{
        int nstns= nlops+1;
        int range_ptr[MAX_STATIONS];
        int  i , j , k, i2;

        /* LOP pair selection process */
        int LOP_4[] = { 1, 3,
                        3, 5,
                        5, 2,
                        2, 4,
                        4, 1};
        int LOP_3[] = { 1, 3,
                        2, 3,
                        2, 4};
        int LOP_2[] = { 1, 2,
                        2, 3};
        double  min_snr, max_snr;
        int     min_snr_pos, max_snr_pos;

        for (i=0; i < nstns; i++)
                range_ptr[i] = i;

/*
** --- RE-ORDER FOR THE OMEGA BEARING ARRAY (ABOMINABLE SORT) WITH
**      DE-REFERENCING.
*/
#if DEBUG
        printf(" MODULE enumerate_LOPs\n");
        printf("\tBearings:\n");
        for (i=0; i < nstns; i++)
                printf("\t\trange_bearing[%d] = %lf\n",
                        stations[range_ptr[i]],
                        baz[stations[range_ptr[i]]]);

#endif
        for (i=0; i < nstns; i++)
                for (j=i+1; j < nstns; j++)
                {
                        if (baz[stations[range_ptr[i]]] >
                                baz[stations[range_ptr[j]]])
                        {
                                k = range_ptr[j];
                                range_ptr[j] = range_ptr[i];
                                range_ptr[i] = k;
                        }
                }

#if DEBUG
        printf("\n\t SNR:\n");
        for (i=0; i < nstns; i++)
                printf("\t\tR[%d] = %lf\n",
                        stations[range_ptr[i]],
                        R[stations[range_ptr[i]]]);
#endif

        switch(nlops)
        {
        case 4: /* NLOPS = 4 */

                min_snr = UNKNOWN_TRACE;        /* Some large value */

                for (i=0; i < nstns; i++)       /* Find the smallest SNR position */
                        if (min_snr > R[stations[range_ptr[i]]])
                        {
                                min_snr = R[stations[range_ptr[i]]];
                                min_snr_pos = i;
                        }

                for (i=0; i < nlops; i++)       /* Set up the pairs */
                {
                        i2 = i + i;
                        j = (min_snr_pos + LOP_4[i2] - 1) MOD nstns;
                        k = (min_snr_pos + LOP_4[i2+1] - 1) MOD nstns;
                        pair[i][0] = range_ptr[j];
                        pair[i][1] = range_ptr[k];
                }
                break;
        case 3:
                min_snr = UNKNOWN_TRACE;        /* Some large value */

                for (i=0; i < nstns; i++)       /* Find smallest SNR position */
                        if (min_snr > R[stations[range_ptr[i]]])
                        {
                                min_snr = R[stations[range_ptr[i]]];
                                min_snr_pos = i;
                        }

                for (i=0; i < nlops; i++)       /* Set up the pairs */
                {
                        i2 = i + i;
                        j = (min_snr_pos + LOP_3[i2] - 1) MOD nstns;
                        k = (min_snr_pos + LOP_3[i2+1] - 1) MOD nstns;
                        pair[i][0] = range_ptr[j];
                        pair[i][1] = range_ptr[k];
                }
                break;
        case 2:
                max_snr = -1.;                  /* Some small value */

                for (i=0; i < nstns; i++)       /* Find smallest SNR position */
                        if (max_snr < R[stations[range_ptr[i]]])
                        {
                                max_snr = R[stations[range_ptr[i]]];
                                max_snr_pos = i;
                        }

                for (i=0; i < nlops; i++)       /* Set up the pairs */
                {
                        i2 = i + i;
                        j = (max_snr_pos + LOP_2[i2]) MOD nstns;
                        k = (max_snr_pos + LOP_2[i2+1]) MOD nstns;
                        pair[i][0] = range_ptr[j];
                        pair[i][1] = range_ptr[k];
                }
                break;
        otherwise:
                break;
        }
#if DEBUG
        printf("\n\t PAIR SELECTED:\n");

        for (i=0; i  < nlops; i++)
                printf("\t\t  pair -> %d %d\n", pair[i][0], pair[i][1]);
#endif
}
```

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence Research Establishment Ottawa
Shirley Bay

2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C,R or U) in parentheses after the title.)

Automatic Omega Station and LOP Selection (U)

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)

McMillan, J. Christopher

5. DATE OF PUBLICATION (month and year of publication of document)

Sept. 1987

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)

124

6b. NO. OF REFS (total cited in document)

13

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

DREO Report

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

DMEE-7

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)

DMEE 043

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

DREO REPORT 976

10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify)

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

(U) Omega is a worldwide navigation system based on 8 VLF transmitting stations dispersed widely around the globe. The phase difference between the signals of any two Omega stations can provide a line of position (LOP) on the earth's surface. Two such LOPs intersect to provide a position fix. Generally several LOPs are selected to improve the accuracy and reliability of the fix either by using a least squares technique or as in MINS (Marine Integrated Navigation System) a more sophisticated Kalman filter technique. The resulting position accuracy and reliability depends on many factors such as geometry and signal strength. This report describes these factors and a set of algorithms used by MINS to automatically select a best choice of 5 Omega stations and 4 LOPs. This includes a modal interference predictor and a multi-LOP geometric dilution of precision calculation.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible, keywords should be selected from a published thesaurus e.g. *Thesaurus of Engineering and Scientific Terms* (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

NAVIGATION
OMEGA
MINS
AUTOMATIC
GDOP
MODAL INTERFERENCE

END
DATE
FILMED
DTIC
JULY 88